Pharmacy Inventory Optimization Using Demand Forecast Data: Deep Learning Approach

Dissertation submitted to

Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology

in

Computer Science & Engineering

by

Shreyas Mendhekar (MT23MCS004)

under the guidance of

Dr. R. B. Keskar

Department of Computer Science &Engineering

Visvesvaraya National Institute of Technology

Nagpur 440 010 (India)

JUNE 2025

Pharmacy Inventory Optimization Using Demand Forecast Data: Deep Learning Approach

Dissertation submitted to

Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology

in

Computer Science & Engineering

by

Shreyas Mendhekar (MT23MCS004)

under the guidance of

Dr. R. B. Keskar

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology

Nagpur 440 010 (India)

JUNE 2025

©Visvesvaraya National Institute of Technology (VNIT) 2025

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology, Nagpur

Declaration

I, **Shreyas Mendhekar**, hereby declare that this dissertation titled **"Pharmacy Inventory Optimization Using Demand Forecast Data: Deep Learning Approach"** is carried out by me in the Department of Computer Science & Engineering of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution / University.

Name and signature

Date: 06/06/2025

Certificate

This to certify that the dissertation titled **"Pharmacy Inventory Optimization Using Demand Forecast Data: Deep Learning Approach"** is submitted by **Shreyas Mendhekar** in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science & Engineering,** VNIT Nagpur. The work is comprehensive, complete and fit for final evaluation.

Dr. R. B. Keskar
Associate Professor
Department of Computer Science & Engineering, VNIT Nagpur

सह. प्राध्यापक
Associate Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept of Computer Science & Engg
वि रा प्रौ स नागपुर-10 (भारत)
V.N.I.T. NAGPUR 440010 (India)

Dr. M. P. Kurhekar
Head, Department of Computer Science & Engineering, VNIT Nagpur
Date

विभाग प्रमुख
Head
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg.
वि.रा.प्रौ.सं. नागपुर-१० (भारत)
V.N.I.T., Nagpur-440010. (India)

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology,

Nagpur

Non-Plagiarism Certificate

Certified that M. Tech Dissertation titled **"Pharmacy Inventory Optimization Using Demand Forecast Data: Deep Learning Approach"** submitted by **Shreyas Mendhekar**, Enrollment No. **MT23MCS004**, ID No. **30101** has been checked for plagiarism using TURNITIN software and the overlap is found within prescribed limits. A summary of the report is given below.

MT23MCS004_MTech_Thesis_final.pdf

ORIGINALITY REPORT

12%	8%	9%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

Shreyas Mendhekar

Enrollment No: MT23MCS004.

Dr. R. B. Keskar

Associate Professor

Department of Computer Science &

Engineering VNIT, Nagpur

सह. प्राध्यापक
Associate Prof sso
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept of Computer Science & Engg
वि रा पी सं नागपुर-10 (भारत)

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my guide, Dr. R. B. Keskar, for his exceptional guidance, continuous encouragement, and insightful feedback throughout the course of this project, *"Pharmacy Inventory Optimization Using Demand Forecast Data: Deep Learning Approach."* His expert supervision, patience, and unwavering support have been instrumental in shaping the direction and outcome of this work. I am truly fortunate to have had the opportunity to work under his mentor-ship, which has deeply enriched my learning experience.

I am also grateful to all the faculty members and non-teaching staff of the Computer Science and Engineering Department for their support, cooperation, and for providing the essential resources required to complete this project successfully.

Finally, I would like to extend my sincere thanks to my parents, whose constant encouragement, unconditional love, and belief in me have been my greatest strength throughout this journey.

ABSTRACT

This project addresses the problem of demand forecasting for pharmacy, with a specific focus on its critical role in inventory optimization within supply chain management. This work emphasizes improving forecasting accuracy as part of inventory optimization. . The ultimate objective of this work is to develop a comprehensive solution that utilizes forecasted demand data to optimize inventory levels minimizing inventory costs, improving storage space utilization, and reducing stockouts.

Pharmaceutical sales data, consisting of time series data for eight different medicine classes, is used for demand forecasting. Exploratory Data Analysis (EDA) is performed to examine the components of the time series data. Advanced Deep Learning models such as LSTM, Stacked LSTM, Bidirectional LSTM, and the transformer-based Temporal Fusion Transformer (TFT) are applied for forecasting. A comparative analysis of these models is carried out using evaluation metrics such as Mean Squared Error (MSE) and Systematic Mean Absolute Percentage Error (SMAPE).

Optimization function is introduced which mainly uses the demand forecasted data for giving the output as optimal order quantity for a medicine. Solving the optimization problem requires some generated data for realistic demand data and other constants. For realistic demand generation, TimeGAN is used alongside a customized algorithm specifically designed for generating demand data. The optimization function is solved with the Python library, which gives optimal order quantity as output.

TABLE OF CONTENTS

Title **Page No.**

ABSTRACT ..... **V**

TABLE OF CONTENTS ..... **VI**

LIST OF TABLES ..... **IX**

LIST OF FIGURE ..... **X**

1. INTRODUCTION **1**

1.1 General Background 1

1.2 Demand Forecasting And Inventory Optimization 2

1.3 Objectives 2

1.4 Methodology 3

1.4.1 Demand Forecasting And Comparative Analysis 3

1.4.2 Inventory Optimization Based On The Demand Forecasted Data 6

1.5 Thesis Outline 7

2. LITERATURE SURVEY **9**

2.1 Demand Forecasting Using Machine Learning Techniques 9

2.2 Transformer For Demand Forecasting 11

2.3 GAN Based Time Series Data Generation 11

2.4 Inventory Optimization 12

3. TECHNICAL BACKGROUND **13**

3.1 LSTM (Long Term Short Term Memory) 13

3.2 Stacked LSTM 14

3.3 BiLSTM (Bidirectional Long Term Short Term Memory) 15

3.4 Temporal Fusion Transformer (TFT) 15

3.4.1 Overview of The Architecture of TFT 16

3.5 Time GAN 16

3.5.1 Architecture of Time GAN 17

4.EXPLORATORY DATA ANALYSIS AND DEMAND FORECASTING ALGORITHMS **18**

4.1 Dataset Description 18

4.2 Exploratory Data Analysis (EDA) 19

4.2.1 Dataset Exploration And Analysis ····· 19
4.2.2 Trends ····· 20
4.2.3 Seasonality ····· 23
4.2.4 Lag Analysis Using Auto Correlation Functions ····· 26
4.2.5 Stationarity Analysis ····· 28
4.3 Data Preparation Sequence Split ····· 29
4.4 Deep Learning Based Algorithms ····· 29
4.4.1 Long Term Short Term Memory(LSTM) ····· 30
4.4.2 Stacked LSTM ····· 31
4.4.3 Bi-Directional LSTM ····· 31
4.4.4 Temporal Fusion Transformer (TFT) ····· 32
4.5 Evaluation Matrices ····· 32
4.5.1 MSE (Mean Squared Error) ····· 32
4.5.2 SMAPE (Systematic Mean Absolute Percentage Error) ····· 33
4.5.3 RMSE (Root Mean Squared Error) ····· 34
5. INVENTORY OPTIMIZATION WITH DEMAND FORECASTED DATA ····· 35
5.1 Optimization Problem Definition ····· 35
5.1.1 Problem Definition ····· 35
5.1.2 Objectives ····· 35
5.2 Objective Function ····· 36
5.2.1 Inventory Stocks Considering Demand And Pending Orders ····· 36
5.2.2 Constraints ····· 37
5.2.3 Decision Variable ····· 38
5.3 Realistic Demand Data Generation ····· 38
5.3.1 Criteria For Realistic Demand Data Generation: ····· 38
5.3.2 Customized Algorithm For Generate Medicine Demand (Start_Date, End_Date): ····· 39
5.3.3 Observation of Customized Algorithm ····· 41
5.4 Time Gan For Data Generation ····· 43
5.4.1 Model Training And Results ····· 43
5.5 Optimization Problem Solving With Pulp ····· 45
5.5.1 Optimization On Sample Data ····· 45
5.5.2 Optimized Result ····· 46
6. RESULT AND ANALYSIS ····· 50
6.1 Comparative Analysis of Algorithms ····· 50
6.1.1 LSTM ····· 51

6.1.2 Stacked LSTM ····· 51
6.1.3 Bidirectional LSTM ····· 51
6.1.4 Temporal Fusion Transformer (TFT) ····· 51
6.2 Poor Performance Of TFT ····· 51
6.3 Comparative Analysis ····· 52
7. FUTURE WORK AND CONCLUSION ····· 53
7.1 Conclusion ····· 53
7.2 Limitations ····· 53
7.3 Future Works ····· 54
BIBLIOGRAPHY ····· 55

LIST OF TABLES

5.1 Input sample data for optimization problem 46
5.2 Optimized order quantity 49
6.1 MSE 50
6.2 SMAPE..... 50

LIST OF FIGURE

1.1 Flowchart for comparative analysis of Deep Learning models..... 4
1.2 Flowchart for Inventory optimization based on the demand forecasted data..... 6
3.1 LSTM Architecture..... 13
3.2 Bidirectional LSTM Architecture..... 15
4.1 Weekly sales analysis..... 19
4.2 Monthly sales analysis..... 20
4.3 Trend analysis with rolling mean 1..... 21
4.4 Trend analysis with rolling mean 2..... 21
4.5 Trend analysis with 365 days rolling mean..... 23
4.6 Seasonality 1..... 24
4.7 Seasonality 2..... 25
4.8 ACF graph..... 26
4.9 PACF graph..... 27
5.1 Result of customized demand generation algorithm..... 41
5.2 Generated data with customized algorithm 1..... 42
5.3 Generated data with customized algorithm 2..... 42
5.4 Data distribution of generated data..... 42
5.5 Generated data with Time GAN 1..... 44
5.6 Generated data with Time GAN 2..... 44

NOMENCLATURE

LSTM	Long Term Short Term Memory
TFT	Temporal Fusion Transformer
EDA	Exploratory Data Analysis
GAN	Generative Adversarial Network
ACF	Autocorrelation Function
PACF	Partial Autocorrelation Function
RNN	Recurrent Neural Network
MSE	Mean Squared Error
SMAPE	Systematic Mean Absolute Percentage Error.
ROP	Reordering Point
SS	Safety Stock
hi	Holding Cost
ci	Ordering cost
Z	Total Cost
Oi	Order quantity
Ed	Exceeding demand
L	Lead time
K	Service Level factor
LPP	Linear Programming.
ATC	Anatomical Therapeutic Chemical

Chapter 1

INTRODUCTION

1.1 General Background

The Pharmaceutical industry is the backbone of the global healthcare industry. It plays an important role in enhancing the quality of life, increasing life expectancy and providing quality medications. This industry contributes to public health through research, production, life saving vaccines and medications. It has an impact on the healthcare sector, economy, research and development of any country. Currently the global market of pharmaceutical industry is about 1.5 trillion dollars.

In addition to the health-care and medical technology the pharmaceutical industry plays a vital role in emergency preparedness such as a sudden outbreak of a disease. COVID-19 pandemic was a disaster for the pharmaceutical industry supply chain, as during this period the demand for surgical equipment and medicines was very high. Due to the inefficient management of pharmaceutical inventory, many countries faced the shortage of necessary medicines and equipment. Supply chain management is important in any industry, specifically in the industries that are dealing with the essential products. Inventory management is one of the important parts of the supply chain management life cycle of a product. It primarily focuses on the flow of raw materials to finished goods within warehouses, including *Maintenance, Repair, and Operating (MRO)* supplies. Effective inventory management leads to an increased revenue, reduced cost, improved customer service, reduced waste and risk mitigation.

In the pharmaceutical industry, the inventory includes different kinds of medicines including those which require certain storage conditions, like temperature, some medicine with shorter expiry date etc. To manage all the inventory of different products

to achieve cost effective space optimization and to reduce wastage, requires a perfect inventory management technique to satisfy the constraints.

This study mainly focuses on the demand forecasting along with how we can use this demand forecast data effectively for inventory optimization.

1.2 Demand Forecasting And Inventory Optimization

Demand forecasting is the process of prediction of the future demands of a product or service using historical sales data, seasonality of the product, market overall trend etc. This kind of data can be seen as time series data, which is a sequence of data points over a period of time or over certain intervals like on hourly, daily, weekly basis.

There are different methods and [1] techniques for demand forecasting

- ABC analysis
- JIT (Just in Time)
- SKU Rationalization
- Demand forecasting.

The inventory optimization aims to minimize total inventory costs such as holding, ordering, and stock-out costs, while ensuring that customer demand is met without delays. By integrating accurate demand forecasting with inventory optimization [2] strategies, businesses can streamline the movement of goods from raw materials to finished products, including essential *Maintenance, Repair, and Operating (MRO)* supplies. This integration not only reduces excess inventory and prevents stock shortages but also enhances service levels, improves cash flow and supports overall supply chain agility.

This study is based on the pharmaceutical sales dataset which is used for time series analysis and for demand forecasting purposes.

1.3 Objectives

The primary objective of this study is to analyze and compare the performance of various deep learning algorithms for time series forecasting of pharmaceutical demand data pertaining to eight different medicines, and to identify the most effective model for accurate demand prediction.

The sub-objectives of this study, are as follows

- **Demand forecasting using Deep Learning Algorithms**

1. To collect and analyze the time series pharmaceutical sales data of eight selected medicines.
2. To perform exploratory data analysis (EDA) for understanding the patterns, trends, and seasonality in the time series data.
3. To explore and implement deep learning-based algorithms for pharmaceutical demand forecasting.
4. To investigate Transformer-based algorithms specifically designed for time series forecasting.
5. To perform hyper-parameter tuning of the selected algorithms to improve forecasting accuracy.
6. To conduct a comparative analysis of the implemented algorithms based on standard evaluation metrics

- **Inventory Optimization Based On Demand Forecasted Data**

1. To define optimization problems for optimizing the order quantity for the inventory based on the demand forecasted data.
2. To define the optimization function and the constraints.
3. To generate the required data for solving the optimization problem.
4. To explore the GAN algorithm for realistic demand data generation.
5. To solve the optimization problem using the Python library.

1.4 Methodology

The methodology of this study is structured into two main steps:

Step1 : Demand forecasting and Comparative analysis.

Step 2: Inventory optimization using demand forecasted data.

1.4.1 Demand Forecasting And Comparative Analysis

In this work, we have followed following steps for demand forecasting:

1. Analyze the time series data for understanding the important insights like trends, seasonality and stationarity .

2. Apply the LSTM variants and Transformer based algorithms for demand forecasting.

3. Hyper-parameter tuning of algorithms for better performance.

4. Comparative analysis of the algorithms used.

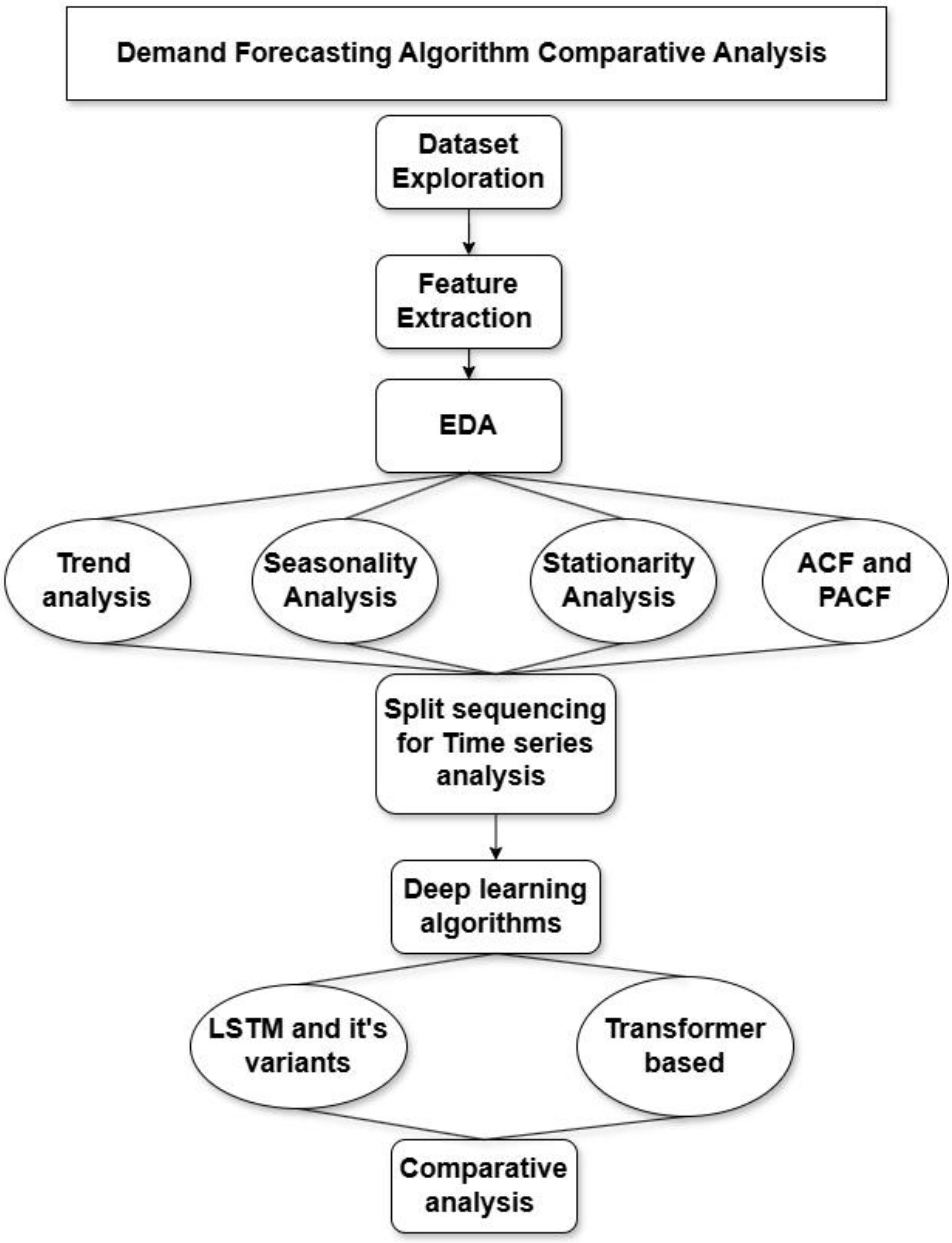

Figure 1.1: Flowchart for comparative analysis of Deep Learning models.

The flowchart represents the step-by-step process followed for the comparative analysis of deep learning algorithms for demand forecasting of pharmaceutical sales data.

1. Dataset Exploration

The process begins with exploring the pharmaceutical sales dataset to understand its structure, attributes, and quality.

2. Feature Extraction

Relevant features are extracted from the dataset to ensure the data is suitable for time series forecasting.

3. Exploratory Data Analysis (EDA)

EDA is conducted to gain insights into the data and identify key characteristics necessary for modeling.

Detailed Analysis Components of EDA

- **Trend Analysis:** Identifies long-term patterns or directions in the data.
- **Seasonality Analysis:** Detects repeating short-term cycles or patterns.
- **Stationarity Analysis:** Determines whether statistical properties of the series (like mean and variance) remain constant over time.
- **ACF and PACF Analysis:** Autocorrelation (ACF) and Partial Autocorrelation (PACF) functions help in identifying relationships within the time series data, useful for selecting appropriate models.

4. Split Sequencing for Time Series Analysis

The dataset is split into sequences (training, validation, test) for model development and evaluation.

5. Deep Learning Algorithms

Two major categories of algorithms are applied for forecasting:

- **LSTM and its Variants:** Long Short-Term Memory networks and their modified forms, suitable for capturing temporal dependencies.
- **Transformer-Based Models:** Advanced architectures that leverage attention mechanisms for efficient time series forecasting.

6. Comparative Analysis

Finally, the performance of the implemented algorithms is compared using suitable evaluation metrics to identify the most effective forecasting model.

1.4.2 Inventory Optimization Based On The Demand Forecasted Data

In this work, we have followed following steps for Inventory Optimization based on the demand forecasted data:

1. Define the optimization problem with proper constraints and decision variables.
2. Realistic demand and required data generation for solving the optimization problem.
3. Apply the optimization solver to solve the problem and find the optimized order quantity.

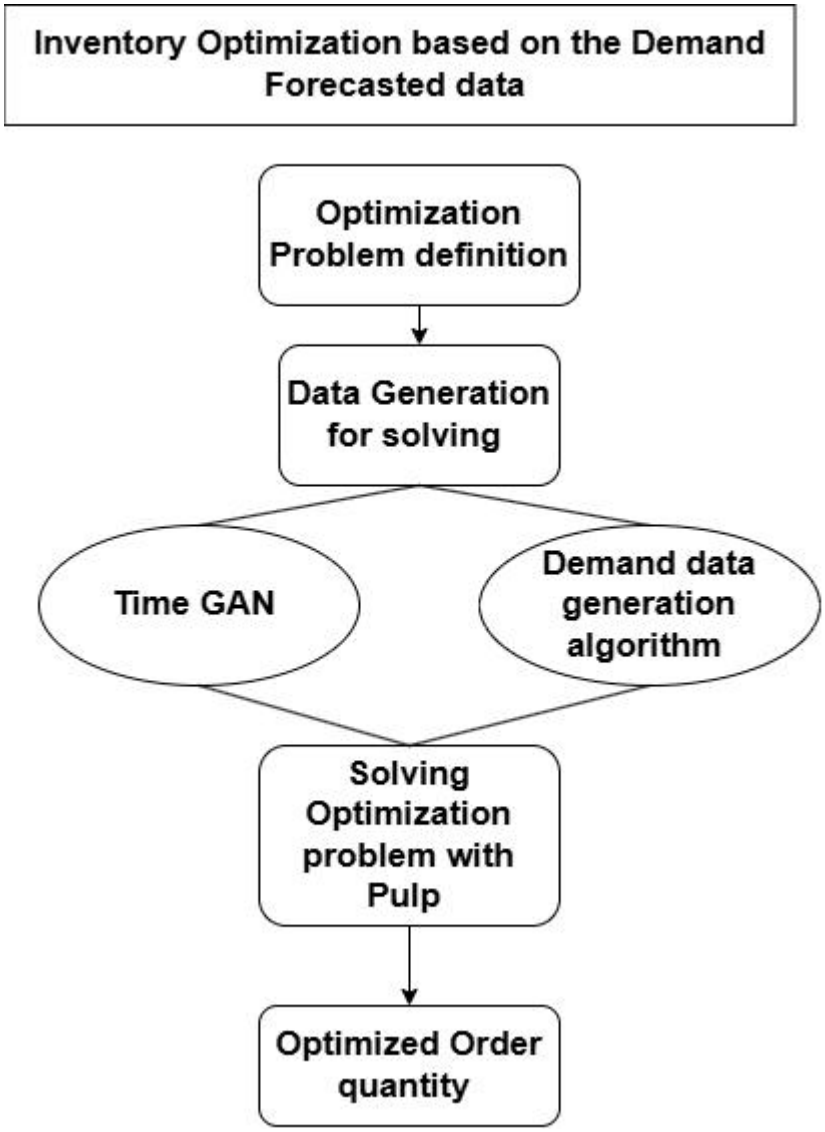
Inventory Optimization based on the Demand Forecasted data
Optimization Problem definition
Data Generation for solving
Time GAN
Demand data generation algorithm
Solving Optimization problem with Pulp
Optimized Order quantity

Figure 1.2: Flowchart for Inventory optimization based on the demand forecasted data

This flowchart illustrates the process of optimizing inventory based on the forecasted demand data, using both synthetic and algorithm-generated demand scenarios.

1. Optimization Problem Definition

The process begins with formulating the inventory optimization problem, including defining the objective function (e.g., minimizing total cost) and identifying decision variables and constraints.

2. Data Generation for Solving

To solve the optimization problem, realistic demand data is required. This step focuses on generating the required data for the optimization model.

3. Demand Data Generation Approaches:

- **TimeGAN:**

A generative adversarial network specifically designed for generating synthetic time series data, used here to generate realistic demand scenarios when limited real data is available.

- **Customized Demand Data Generation Algorithm:**

Custom algorithms or forecasting models are used to generate forecasted demand data based on historical pharmaceutical sales.

4. Solving the Optimization Problem with PuLP

Using PuLP, a linear programming optimization library in Python, the formulated problem is solved to compute the optimized order quantities for each medicine.

5. Optimized Order Quantity

The output of the process is the optimal order quantity for each pharmaceutical product, ensuring minimized inventory costs and fulfillment of demand based on the generated data.

1.5 Thesis Outline

This thesis is organized as follows:

Chapter 1: Introduction

This chapter outlines the problem definition, presents the current state of research on the topic, and introduces the methods and overall methodology adopted to address the problem.

Chapter 2: Literature Survey

This chapter provides a comprehensive review of existing research and related works that have informed and guided the development of this project.

Chapter 3: Technical Background

This chapter presents the underlying principles, working mechanisms, and architectural details of the deep learning algorithms utilized in this study.

Chapter 4: EDA and Demand forecasting algorithms

This chapter discusses the essential time series analysis methodologies employed for demand forecasting, along with the deep learning neural network architectures and their corresponding hyperparameters.

Chapter 5: Inventory optimization based on demand forecasted data

This chapter provides the optimization function definition and the necessary demand data generation techniques for solving the problem.

Chapter 6: Results and Analysis

This chapter presents a comparative analysis of the deep learning algorithms used for demand forecasting, along with the results obtained from the inventory optimization problem.

Chapter 7: Conclusion and Future Work

This chapter additionally provides conclusions drawn from results and mentions ideas for further research.

Chapter 2

LITERATURE SURVEY

This chapter presents the background work done by researchers in the area of demand forecasting in inventory management / supply chain management.

2.1 Demand Forecasting Using Machine Learning Techniques

In 2022, a study published in the *International Journal of Advances in Engineering and Pure Sciences* by **Salih İmece** and colleagues [3] investigated demand forecasting challenges within the pharmaceutical sector, focusing on wholesaler order prediction. Their approach combined time series analysis with machine learning models to forecast the demand of a single pharmaceutical product. The techniques applied included Holt-Winters, Ridge Regression, Random Forest, and XGBoost. To enhance forecasting precision, ensemble methods were employed by merging the outputs of these individual models. Performance evaluation using the Root Mean Square Percentage Error (RMSPE) indicated that ensemble methods outperformed standalone models. The researchers emphasized the importance of expanding future work to include a broader range of pharmaceutical items and suggested the use of deep learning and data mining techniques for improved outcomes.

In another study published in *Supply Chain Forum*, **Raheel Siddiqui et al**. [4] proposed a hybrid demand forecasting model aimed at improving accuracy in pharmaceutical supply chains. They introduced ARHOW, a model combining Auto-Regressive Integrated Moving Average (ARIMA) with Holt-Winters (HW) smoothing. The methodology involved generating forecasts using both ARIMA and HW, then combining

them using a weighted model equation. The Mean Absolute Percentage Error (MAPE) was used to evaluate performance. The ARHOW model demonstrated superior accuracy with a MAPE of 1.99%, significantly better than ARIMA (5.64%), Holt's (9.37%), ETS (6.33%), and Theta (6.21%). These results highlighted the benefits of hybrid modeling in demand forecasting.

A data-driven strategy for pharmaceutical demand forecasting was presented by **Gunadi Emmanuel, Yulyani Arifin, and co-authors** [5] in a 2023 *IEEE publication*. Their study aimed to improve inventory control by integrating epidemiological variables—such as disease transmission rates, weather conditions, and demographic features like age and gender into the forecasting process. These variables were processed using a Random Forest algorithm to make time-based predictions. The results indicated that the inclusion of health and environmental data significantly enhanced forecasting performance compared to traditional methods, reinforcing the value of context-aware models in pharmaceutical demand planning.

Rathipriya, Abdul Aziz Abdul Rahman, and their team [6], in a 2022 *Springer publication*, examined the use of neural networks for predicting pharmaceutical demand based on time series data. The study compared various machine learning models, including shallow neural networks—like the Probabilistic Neural Network (PNN), Generalized Regression Neural Network (GRNN), and Radial Basis Function Neural Network (RBFNN) alongside deep learning models such as LSTM and Stacked LSTM. The findings revealed that no single model was universally optimal across all drug categories. The authors concluded that model selection should depend on dataset characteristics and recommended future exploration of hybrid techniques that combine machine learning and traditional forecasting models for better results.

In a 2019 study published by *Springer*, **Amalnick, Habibifar, and colleagues** [7] developed an intelligent framework for forecasting pharmaceutical product demand. Their method utilized a combination of clustering, classification, and neural network modeling. The dataset was first segmented into distinct clusters, with each cluster assigned a customized neural network architecture for forecasting. During prediction, incoming data was classified into one of these clusters, and the corresponding pre-trained

model was used for demand prediction. This modular approach allowed the model to account for product-level variation, enhancing forecast precision and aiding strategic inventory decisions. The study recommended future enhancements for generalization and robustness.

2.2 Transformer For Demand Forecasting

In a 2024 study, **A. Ouaddi, J. Benhra,** and colleagues [8] investigated the effectiveness of Transformer-based models for forecasting customer demand. The researchers conducted a comparative analysis between the Transformer architecture and two widely used deep learning models for time series prediction Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. Unlike traditional models, the Transformer leverages an attention mechanism originally designed for natural language processing tasks. When applied to time-series data, this mechanism allows the model to capture and retain temporal dependencies more effectively. The study's results demonstrated that the Transformer outperformed both CNN and LSTM models, particularly in recognizing long-term patterns in demand behavior. These findings suggest that Transformer-based models hold significant promise for enhancing the accuracy and efficiency of demand forecasting and inventory management strategies.

2.3 GAN Based Time Series Data Generation

Brophy, Wang, Ward, and co-authors [9] investigated the use of Generative Adversarial Networks (GANs) for generating time series data, expanding their traditional applications beyond image synthesis. Their research focused on how GANs can effectively address common issues in time series analysis, such as missing values and limited data availability. The study highlighted GANs' capability to produce synthetic, yet realistic and privacy-preserving, temporal datasets. The authors offered an in-depth classification of time series data, distinguishing between continuous and discrete types, and further categorizing them based on their data values (e.g., real-valued or integer) and structural dimensions (univariate or multivariate). Unlike conventional models that typically generate one data point at a time, GAN-based approaches demonstrated the ability to produce entire sequences while preserving temporal relationships, making them more suitable for complex sequential data generation tasks.

2.4 Inventory Optimization

Li, Arnold, and their collaborators [10] proposed an innovative, data-centric framework for forecasting red blood cell (RBC) demand and optimizing inventory levels. Their research developed a hybrid forecasting model that combines elements of statistical analysis with machine learning methods. Despite its relatively simple design, the hybrid model demonstrated forecasting accuracy on par with more advanced deep learning architectures such as LSTM. The demand predictions produced by this model were then incorporated into a multi-period inventory optimization strategy to determine the most effective RBC ordering schedules either daily or semi-weekly. The study's findings showed that this integrated approach significantly improved inventory efficiency and reduced operational costs compared to conventional inventory management methods used in hospitals.

In this chapter we took an overview of different approaches for demand forecasting tasks, transformer based methods, and GAN for time series data generation. The next chapter aims to develop a technical background consisting of algorithms and neural network models used in this study.

Chapter 3

TECHNICAL BACKGROUND

This chapter focuses on the algorithms used in this study for experimentation of demand forecasting and data generation. The deep learning based algorithms for time-series predictive analysis are discussed in this section. In deep learning, specifically the LSTM based algorithms and their variants along with the transformer based algorithms are discussed.

3.1 LSTM (Long Term Short Term Memory)

LSTM is a type of Recurrent Neural Network (RNN). It can be described as the *RNN with the gated mechanism.* Simple RNN faces the problem of vanishing gradient, which is handled by the RNN with the gating structure. The gating structure controls the flow of the information of the features over the period of time. The predictive analysis of time-series data requires the extraction of long term dependencies of the temporal data. Due to the gating mechanism, the LSTM preserves the long term dependencies effectively. The gates used in the LSTM are described below,

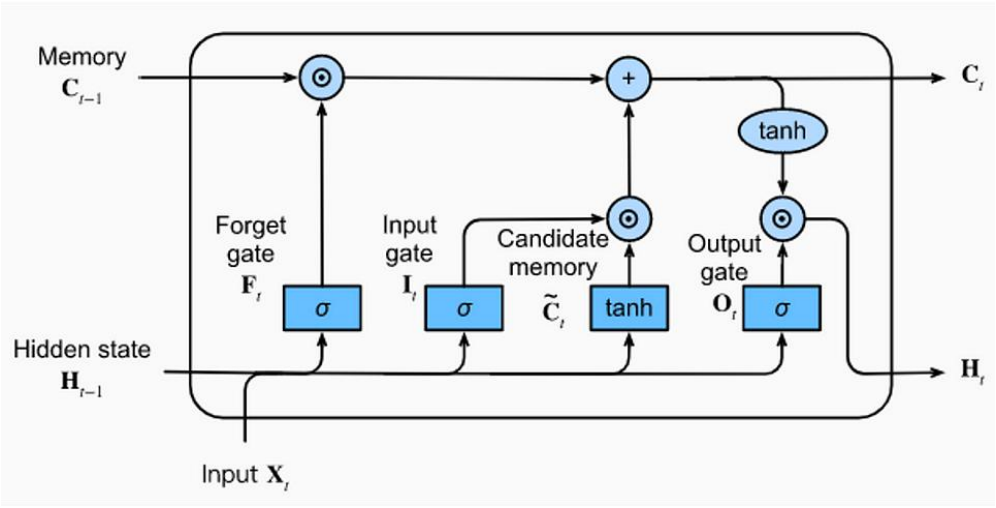

Figure 3.1: LSTM Architecture

Mathematically the LSTM Gate is represented as,

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(Forget Gate)}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(Input Gate)}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad \text{(Candidate Cell State)}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad \text{(New Cell State)}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(Output Gate)}$$
$$h_t = o_t * \tanh(C_t) \quad \text{(Hidden State / Output)}$$

- **Forget Gate:** Decides what information to forget from the cell state C(t-1).
- **Input Gate:** Decides what new information to store in the cell state.
- **Output Gate:** Decides the final output based on the cell state and input.
- **Candidate Cell State:** New candidate values that could be added to the cell state.
- **New Cell State:** Update the cell memory by combining old and new information.
- **Memory Cells:** Store information over long periods.

The working of LSTM cells is as follows: At each time step t, the memory cell receives the input x(t) and hidden state h(t-1) from the previous cell. LSTM follows the recurrent model where each cell receives the input from the previous cell and also the input features. The gate of the memory cell calculates what information to keep or discard and update the current cell C(t). The output h(t) is calculated by the output gate as input for the next cell. In this way LSTM preserves the long term dependencies due to the structure of the gate. Stacked LSTM and BiLSTM have a similar architecture of a Memory cell but slight difference is in the arrangement of the memory cells.

3.2 Stacked LSTM

Stacked LSTM is a variation of the LSTM. This variant is *LSTM with stacking of multiple similar LSTM layers,* which add more complexity and makes the model more deeper. The added layers not only capture the short term dependencies but also the long term dependencies as compared to the single LSTM. The structure of the memory cell is the same as LSTM as explained in the LSTM architecture.

Each layer captures the complex abstract information and transmits it to the next layer, allowing the model to learn the time dependent features more effectively.

3.3 BiLSTM (Bidirectional Long Term Short Term Memory)

BiLSTM is another variant of LSTM where it can process the data in both forward and backward directions. This bidirectional mechanism can consider the past information and future information for the prediction purpose if and only if the entire sequence is available.

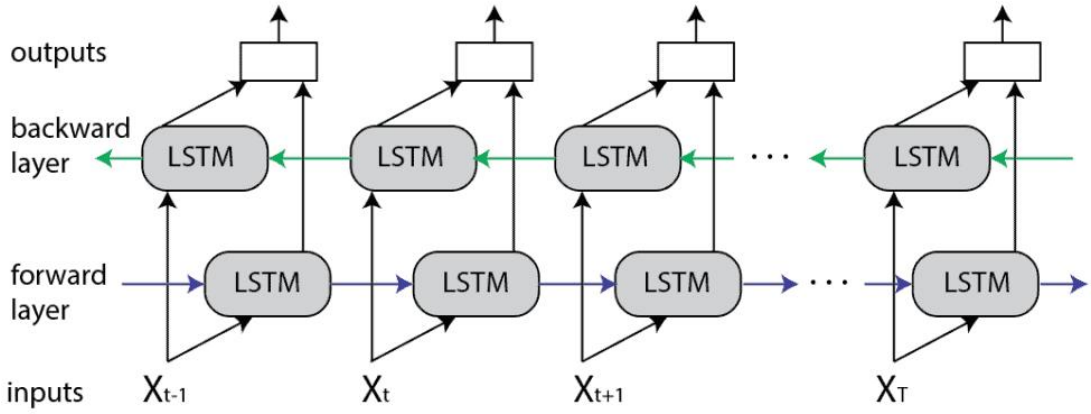

Figure 3.2: Bidirectional LSTM Architecture

BiLSTM has two LSTM layers as shown in the figure 3.2, first layer is forward layer which is similar to the simple LSTM where the feature data points flow in one direction, that is start to end. The second layer is a backward layer which is the reverse of the forward layer i.e. the feature data will flow from the end to the start.

The forward layer mainly captures the long term dependencies of past to the current state, but the backward layer captures the long term dependencies of future to the current state In this way, the BiLSTM have both the dependencies from past as well as from future. This ability of BiLSTM makes it a more complex and more deep neural network, which can extract a complex pattern and long term dependencies for time-series data.

3.4 Temporal Fusion Transformer (TFT)

The Temporal Fusion Transformer (TFT) [11] is a state-of-the-art deep learning model designed for time series forecasting. The ability of TFT that makes it special is the ability to handle both static (features that don't change over the time) and time-varying input features at the same time focusing on the time steps as well as most important

features during prediction. It is the modification of the transformer by incorporating the attention mechanism for capturing the dependencies, gated mechanism and variable selection network. This makes TFT flexible and more accurate. It not only delivers accurate forecasts but also shows which factors contribute in its predictions the most. TFT's ability to capture both short and long term dependencies while focusing on the important responsible features for the prediction makes it one of the most powerful tools available for the time series modeling. It supports the multi-horizon time series forecast, which means multi-step prediction for univariate or multivariate time series data.

3.4.1 Overview of The Architecture of TFT

- **Input Embedding:** Converts the time dependent features and static features into the n-dimensional feature space.
- **Variable Selection Network:** Selection of relevant variables at each time step with the help of attention mechanism.
- **Gated Residual Network:** It is used for skip connection for stability and feature processing using gating.
- **Static co-variate Encoders:** It encodes the static features into the context vector for further processing.
- **LSTM Encoder-Decoder:** It focuses on the handling of past and future known inputs in time series forecasting.
- **Temporal Self Attention Layer:** Self attention mechanism which captures the long term dependencies.
- **Interpretable Output Layer:** The output of TFT is multi step forecasts and also provides the interpretability through weights and feature importance.

3.5 Time GAN

Generative modeling of time-series data is a challenging task due to the need to preserve not only the data distribution but also the temporal dependencies between observations. Traditional generative models such as Generative Adversarial Networks (GANs) have achieved remarkable success in image and text generation but there is a struggle to model sequential patterns intrinsic to time-series data. To address this, TimeGAN (Time-series Generative Adversarial Network), proposed by Yoon et al. (2019), introduces a

novel architecture that combines the strengths of GANs and recurrent neural networks (RNNs) to generate high-fidelity synthetic time-series data.

3.5.1 Architecture of Time GAN

Time GAN consists of five key components [12] :

- **Embedder:** An RNN-based network that maps high-dimensional time-series data into a latent space.
- **Recovery:** A decoder that reconstructs the original data from the latent space.
- **Generator:** Produces synthetic latent sequences from random noise vectors.
- **Discriminator:** Tries to distinguish between real and synthetic latent sequences.
- **Supervisor:** A unique component that ensures that the generator learns temporal dynamics by predicting the next-step hidden state.

Together, these components are trained using a composite loss function consisting of:

- Reconstruction loss (autoencoder-style)
- Supervised loss (temporal consistency)
- Adversarial loss (realism of synthetic data)
- Moment matching loss (to align distributions).

This chapter describes the algorithms used for experimentation on time series data and working of these algorithms. The next chapter focuses on the dataset description, exploratory data analysis of the dataset and the experimental study on time-series data.

Chapter 4

EXPLORATORY DATA ANALYSIS AND DEMAND FORECASTING ALGORITHMS

This chapter presents the dataset exploration, Exploratory Data Analysis (EDA) on pharmaceutical dataset and Deep learning algorithms experimentation for comparative analysis.

4.1 Dataset Description

The pharmaceutical sales dataset is used to carry out the EDA and demand forecasting algorithm analysis. The dataset is available on kaggle. [13]

The pharmacy sales dataset is a time-series data which is recorded over a period of 6 years.

It consists of 600,000 transactional data items of eight ATC (Anatomical Therapeutic Chemical) medications classes. ATC is a drug classification system that categorizes drugs by their therapeutic, pharmacological, and chemical properties, as well as the organ or system they affect.

- **M01AB**: Anti-inflammatory and anti-rheumatic products, non-steroids, acetic acid derivatives, and related substances
- **M01AE**: Anti-inflammatory and anti-rheumatic products, non-steroids, propionic acid derivatives
- **N02BA**: Other analgesics and antipyretics, Salicylic acid, and derivatives
- **N02BE/B**: Other analgesics and antipyretics, pyrazolones, and anilides
- **N05B**: Psycholeptic drugs, anxiolytic drugs
- **N05C**: Psycholeptics drugs, hypnotics, and sedative drugs

- **R03**: Drugs for obstructive airway diseases
- **R06**: Antihistamines for Systemic Use

The dataset contains 4 CSV files that consist of hourly, daily, weekly, and monthly sales data. Each file has 9 columns, of which 8 are of ATC classified medicine types, and the column one contains the data time information.

4.2 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) [14] for time-series involves analysis of time dependent features by discovering the pattern, seasonality, and overall trend over the period of time. The time series data have some common patterns like recurring events, cyclic behaviour and seasonality. Stationarity analysis plays an important role in lag analysis which helps in analyzing the temporal dependencies and seasonality. There are different graphs like ACF(Autocorrelation Function), PACF(Partial Autocorrelation Function), and some tests like KPSS (Kwiatkowski Phillips Schmidt Shimp Test) and ADFT (Augmented Dickey-Fuller Test) for stationarity analysis.

4.2.1 Dataset Exploration And Analysis

Analysis of Weekly and Monthly total sales of all the 8 ATC medicine classes is conducted. This analysis gives an overall idea about the demand in each month and on weekdays.

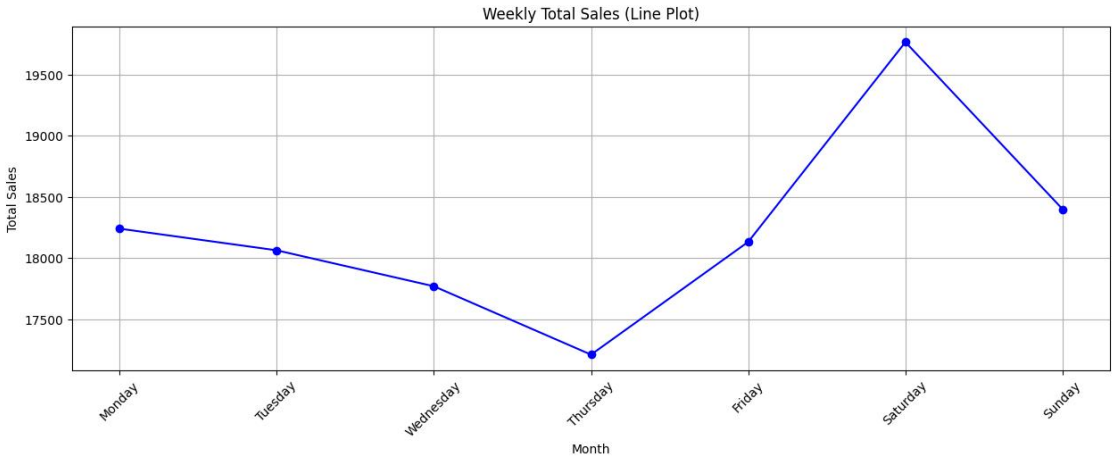

Figure 4.1: Weekly sales analysis.

- **Observations:**
  1. In the weekly sales analysis, the sales on the weekends (mostly on Saturday) are very high as compared to the other weekdays.
  2. Lowest sales on Thursday.
  3. Sales are mostly constant for the remaining weekdays (Monday, Tuesday, Wednesday etc).

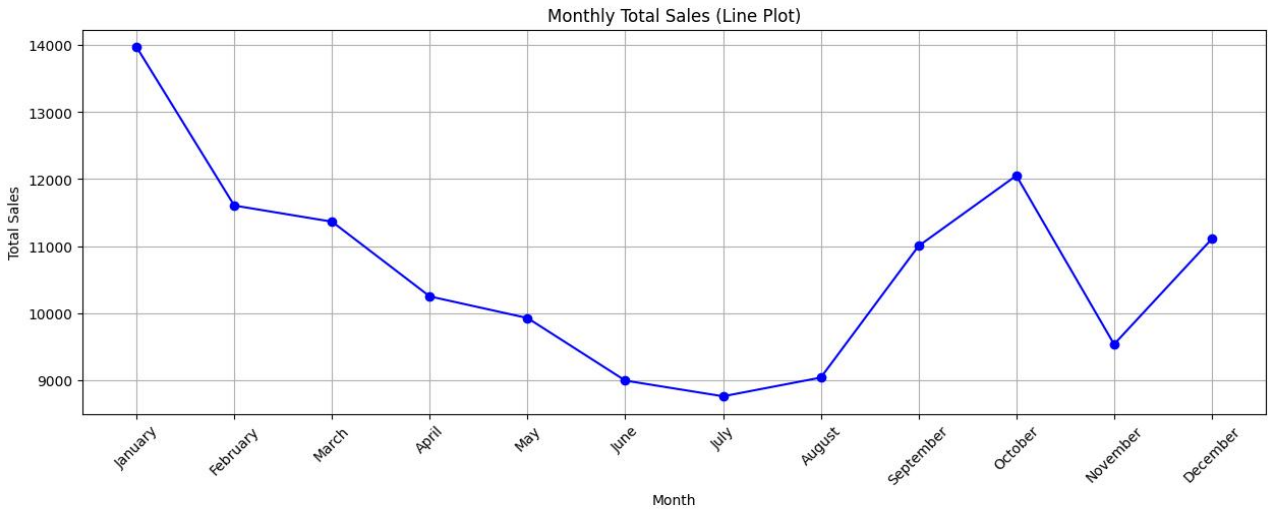

Figure 4.2: Monthly Sales analysis

- **Observations (Total Sales On The Monthly Basis) :**

1. The sales are high during the 1st quarter of the year.
2. In the 2nd & 3rd quarter, the sales are constant.
3. During summer, (March - June), the sales are mostly constant.
4. During the rainy season (July - October) the sales increased slightly compared to the summer.
5. During the Winter season (November - February) overall sales peaked (increased drastically) throughout the winter season.

4.2.2 Trends

Finding the long-term movement or direction in a time series dataset is a crucial step in time series forecasting, and trend analysis is one way to do this. It mainly focuses on whether the data is increasing or decreasing over the period of time. The rolling mean

also known as *Moving Average* is plotted along with the overall daily sales, and rolling mean of 30 days gives the idea about the trend of sales of medication over the months. Similarly the 365 days rolling mean gives the idea about how the trend is moving over the years.

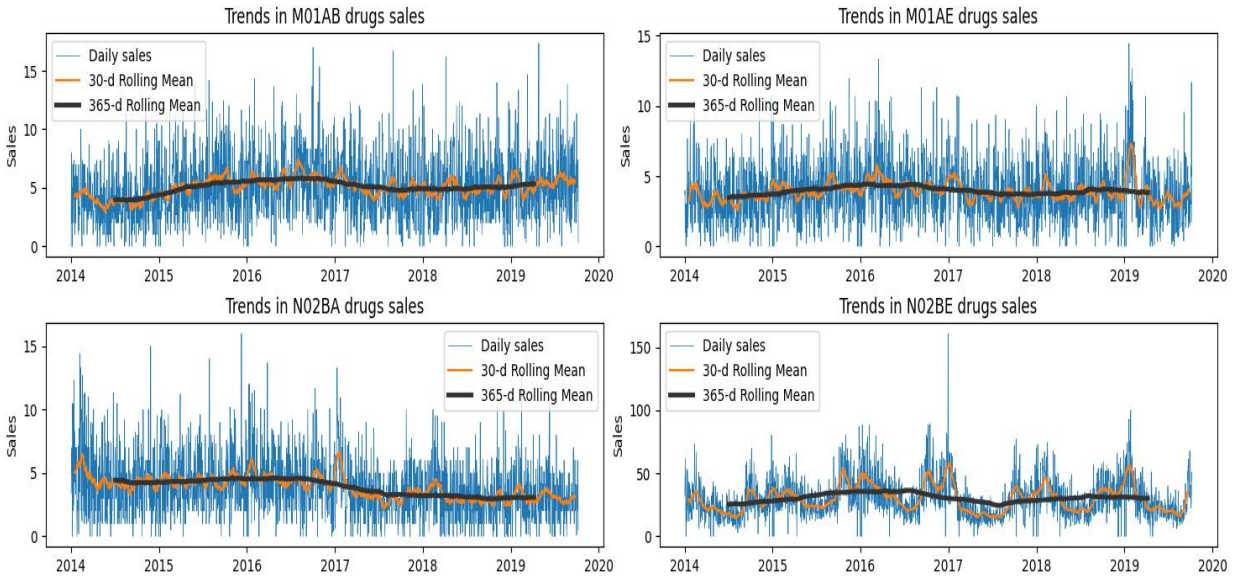

Figure 4.3: Trend analysis with rolling mean 1

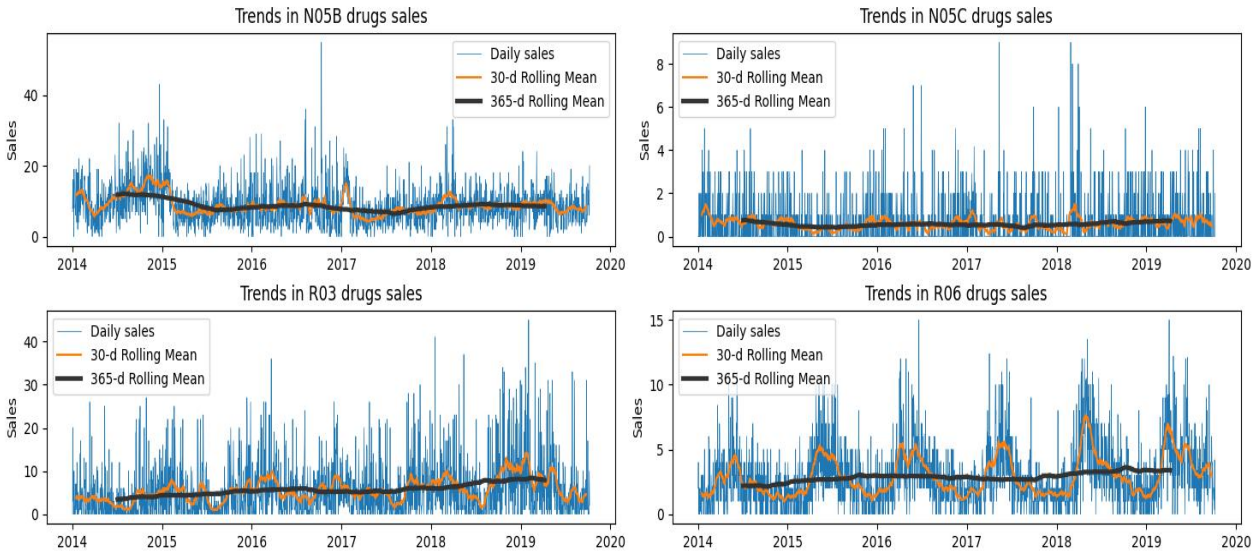

Figure 4.4: Trend analysis with rolling mean 2

The above graphs contains the daily sales of the different classes of medicines and also the rolling mean of 30 days and 365 days. In the figure 4.3 and figure 4.4, trend is

identified and visualized using rolling (moving) averages, which help smooth out short-term fluctuations and highlight long-term patterns.

- **30-day Rolling Mean (Orange Line):**

1. This shows the short-term trend by averaging sales over the past 30 days.
2. It captures seasonal variations and periodic changes in demand.

- **365-day Rolling Mean (Black Line):**

1. This represents the long-term trend by averaging sales over a full year.
2. It helps detect overall growth or decline over multiple years, filtering out seasonal and random noise.

The trend is not computed using a mathematical trend line (like linear regression) but is instead inferred from the slope and shape of the smoothed rolling average lines.

1. Upward Trends:

M01AB and R03: Show a consistent long-term rise in sales, indicating increased demand over time.

2. Downward Trend:

N02BA: Shows a gradual decline, especially after 2015, suggesting reduced usage or market saturation.

3. Seasonal Trends:

R06 and N02BE: Exhibit clear annual cycles with regular peaks, indicating seasonal demand (e.g., related to allergies or flu seasons).

4. Stationary Trends:

M01AE and N05B: Display relatively stable patterns with slight upward movement.

5. Flat/No Trend:

N05C: Sales remain low and fluctuate irregularly, with no clear increasing or decreasing trend.

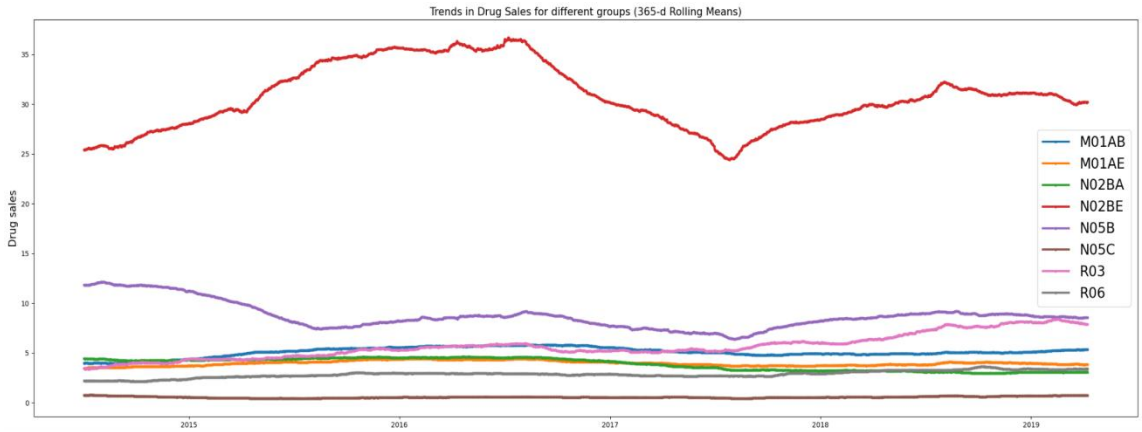

Figure 4.5: Trend analysis with 365 days rolling mean.

Observations on 365-Day Rolling Mean Trends (8 Medication Classes):

1. M01AB - The overall trend is upward/ (Uptrend).
2. M01AE :- Trend is horizontal / stationary in nature.
3. N02BA - Trend is downward (as the sales from mid 2017 are decreasing)
4. N02BE - Trend is horizontal/ stationary in nature.
5. N05B - Trend is horizontal/ stationary in nature.
6. N05C - Trend is mostly stationary or constant (as for many years, the sales are constant)
7. R03 - Over the years, Trend is upwards/ (Uptrend).
8. R06 - The trend is upwards / (Uptrend).

4.2.3 Seasonality

Seasonality analysis in time-series data is a very important component which helps in understanding the recurring patterns or fluctuations, some seasonal factors (increase of sales on holidays, some weather or some festive seasons).

There are different methods for seasonality analysis like seasonal decomposition, box plot (analysis of distribution of data) or ACF (auto correlation function) etc. In this study, the box-plot analysis is used for analyzing the seasonality of 8 different medications mentioned earlier. The interpretations of seasonality based on the box-plot done by analyzing the central tendency like *mean or median* in the distribution of data. Monthly

data of sales are represented in terms of box-plot, below graphs in figure 4.6 and figure 4.7 represents the 12 months data distribution of each medication.

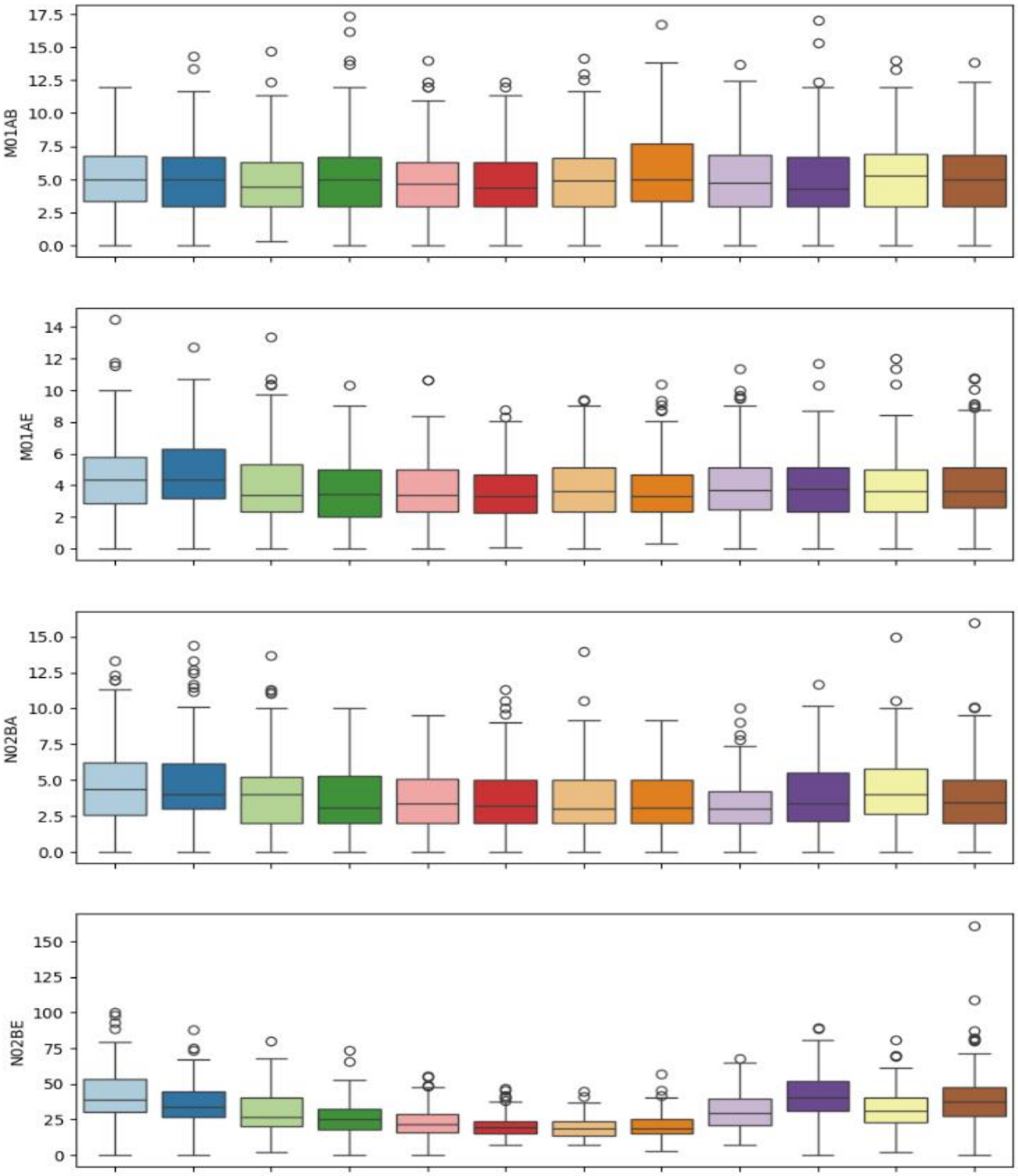

Figure 4.6: Seasonality 1

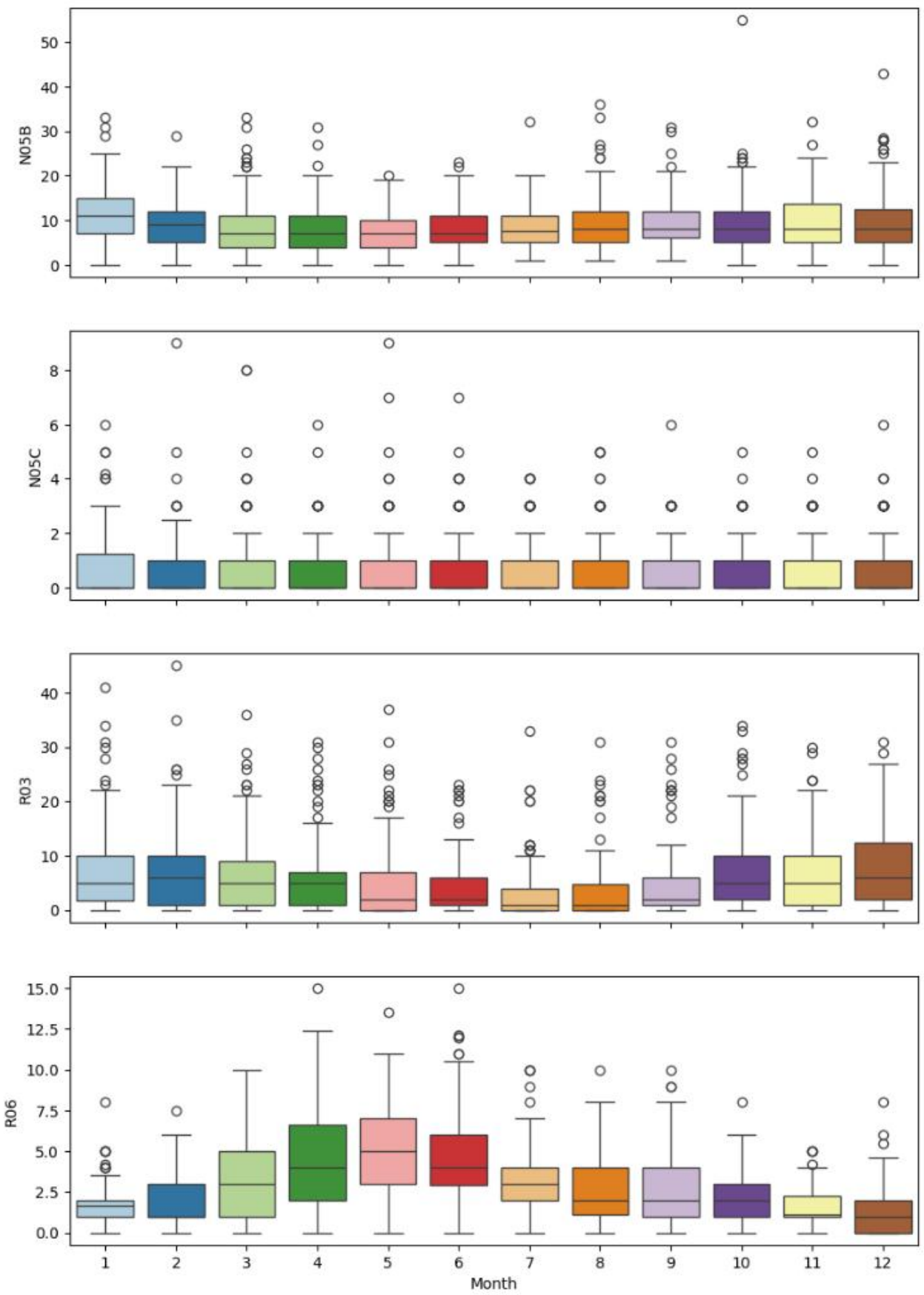

Figure 4.7: Seasonality 2

- **Observations:**

1. **Winter and Early summer Peaks:** Classes like N02BA and R03 have high median between November to February period.
2. **Summer decrease:** Perhaps some classes, like N02BA, have less sales in the period of summer and early monsoon due to lower probability of illness during this season.
3. **Year-End Increase:** Classes such as N02BE, R03, and M01AB, show a rapid rise in demand towards the end of the year (October to December), which might be influenced by colder weather and epidemiological diseases.

4.2.4 Lag Analysis Using Auto Correlation Functions

Lag [15] analysis in time series is the study of how previous (past) values in a series affect current or future values.

- **Autocorrelation Function (ACF):**

ACF measures how similar data points are to each other at different time intervals in a time series. It is the correlation between data points at t(n) and t(n-1) (lag of t(n)). It helps to identify repeating patterns, trends, and seasonality in the data.

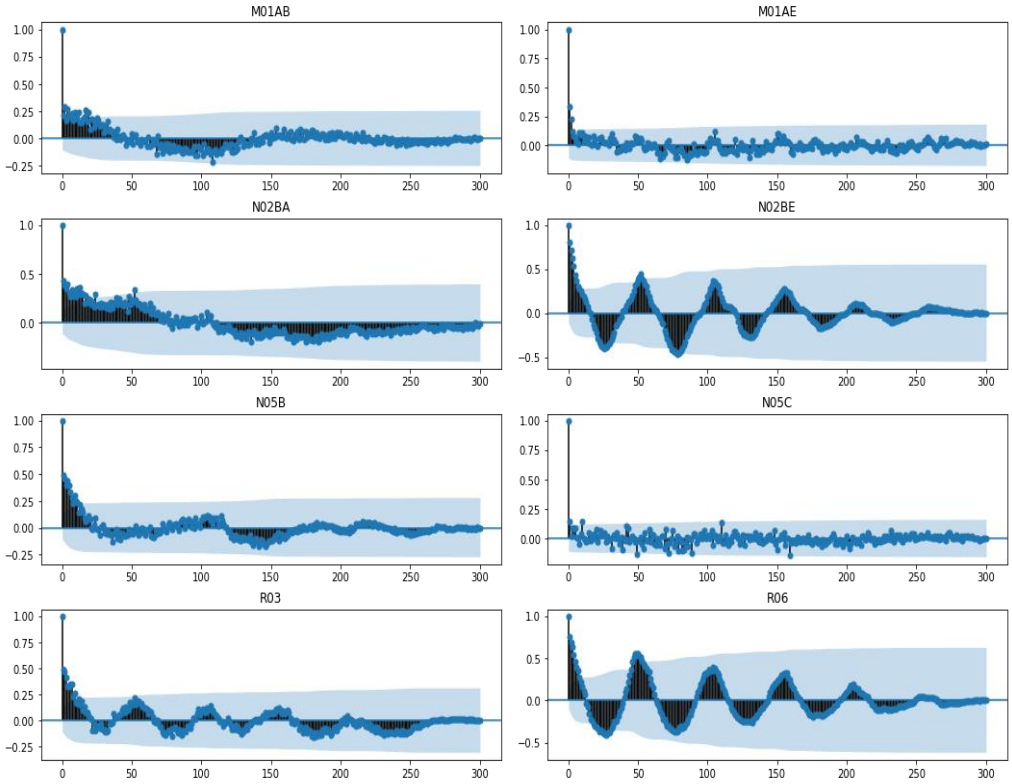

Figure 4.8: ACF graph

- **Observations:**

1. **Stationary Series:** Classes like M01AB, M01AE, N05B, and N05C have stationary behaviour. These classes have short term dependencies and less periodicity.
2. **Seasonal/Periodic Series:** N02BE and R06 shows periodicity, with regular sine waves like seasonality in the ACF.
3. **Mixed Behaviour:** R03 shows the seasonality but not as strongly as N02BE or R06, suggesting the weak periodicity.
4. **Short-Term Dependencies:** All the medication classes have high auto correlation at lag 1, suggesting short-term dependencies are common across the dataset.

- **Partial Autocorrelation Function (PACF):**

PACF measures the correlation between a time series and its lagged values, but removes the effects of the consecutive lags. In simpler terms, it isolates the direct relationship between the time series and a specific lag.

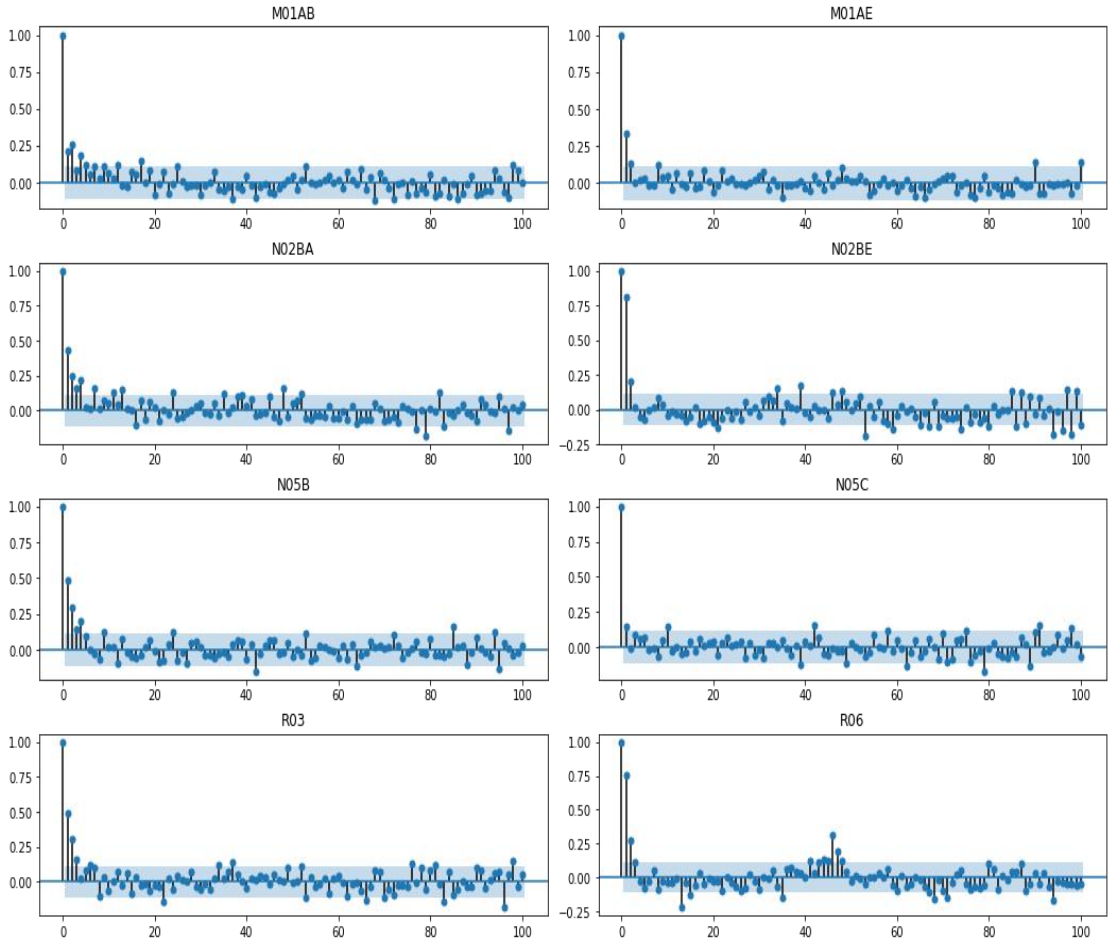

Figure 4.9: PACF graph

- **Observations for PACF:**

1. Short-Term Dependencies: M01AB, M01AE, N05B, and N05C shows the lag one i.e short term autocorrelation dependencies.
2. Higher-Order or Periodic Models: N02BE and R06 show strong periodic spikes and have strong seasonality or periodic dependencies.
3. Mixed behaviour: R03 and N02BA show short-term dependencies with very less influence from higher lags. It implies lag two autocorrelation dependencies.

4.2.5 Stationarity Analysis

The Stationarity analysis plays an important role while working with time-series data. Ideally the time-series data should be stationary. If it is non-stationary then we convert the non-stationarity data into stationary data by differencing or by log transformation. Stationarity analysis is the analysis of the mean, mode, and median of the time-series data. If these parameters change over the period of time, then the time-series is non-stationary, otherwise it is stationary.

There are different tests to determine the stationarity of the model. [16]

1.Augmented-Dickey-Fuller Test (ADF)

The ADF test is used to check whether a time series is stationary (i.e., its statistical properties like mean and variance do not change over time).

- Null hypothesis (H_0): The series has a unit root (i.e., is non-stationary).
- Alternative hypothesis (H_1): The series is stationary.

Interpretation:

- The test statistic is compared to critical values at different confidence levels.
- If the test statistic < critical value, we reject the null hypothesis, indicating stationarity.
- P-value < 0.05 typically supports rejection of the null hypothesis.

2.Kwiatkowski-Phillips-Schmidt-Shin test (KPSS)

- Null Hypothesis (H_0): The time series is stationary around a mean or a trend.
- Alternative Hypothesis (H_1): The series is non-stationary.

Interpretation:

- If the test statistic > critical value, we reject H_0 → non-stationary

- If the test statistic < critical value, we fail to reject H_0 → stationary

- P-value < 0.05 → evidence against stationarity.
- P-value > 0.05 → insufficient evidence to reject stationarity**.**

ADF and KPSS both tests are applied on the dataset, the result of each medication class is given below.

- **Observations of KPSS Test and ADF Test:**

1. According to the ADF test, all the medication classes were found Stationary in nature. It means that there are statistical properties which are constant over the period of time.
2. In KPSS, some classes like R06 and N02BE were found to be non-stationary.

4.3 Data Preparation Sequence Split

Time series data is an ordered data over time, and this temporal structure should be preserved when splitting the data for model testing and development purposes. Random shuffling of data is not suitable for time series forecasting data as it can create information leakage and invalid evaluation. Proper data splitting [17] is essential to ensure realistic model training, validation, and testing that mirrors real-world prediction scenarios.

- **Train-Test Split:** The most basic approach where the large portion of the series is used for training and a smaller portion for testing.

- **Train-Validation-Test Split:** The validation portion is a part of training data which is mainly used in the hyperparameters tuning phase.

- **Sliding Window (split sequence):** This involves fixed step size which is window size of data followed by the next point as output point. In this manner the dataset of time series data is split in sequence of step size.

4.4 Deep Learning Based Algorithms

The algorithms such as LSTM and variants of LSTM are used for demand forecasting. The input data for these neural network models is pharmaceutical sales time-series

dataset. This dataset is split using the split sequencing method. The output of each model is the demand of medication on the next day. This type of prediction is one step forecasting.

4.4.1 Long Term Short Term Memory(LSTM)

LSTM deep learning model is experimented with the pharmaceutical dataset with different parameters. The hyperparameters for LSTM model for time series forecasting of single step prediction are given below:

- **Memory units** - 100.
- **Steps (How many)** - Experimented with - 5- 30 window size.
- **Activation function -** ReLu (Rectified Linear Unit).
- **Optimizers -** Adam.
- **Loss -** MSE (Mean Squared Error) .
- **Epochs -** Experimented with 500 - 1000 epochs.

Memory units are the units which contain the gated structure like input, output, and forgot. Generally 100 units are used when dealing with the large time series dataset; increasing the units will not enhance the performance drastically. The experiments with different window sizes can be done, which represent the split sequencing window sizes. This experiment is to check which window size captures the more temporal dependencies in the time-series data.

Activation function ReLu and Adam optimizer are the two hyperparameters fixed for all algorithms because technically these are best hyperparameters. ReLu solves the problem of vanishing gradient and Adam optimizer helps with dynamic learning rate adjustment for better convergence to the minima. Loss functions used in this study are MSE (Mean Squared Error) , RMSE (Root Mean Squared Error) , and SMAPE (Systematic Mean Absolute Percentage Error) . The epochs being experimented with are in the range of 500-1000.

- **Results After Experimentations:**

After experimentation, the hyperparameters for which the LSTM started converging are epochs at 700 and window step-size of 5. Remaining hyperparameters are untouched as changing them will degrade the performance.

4.4.2 Stacked LSTM

Stacked LSTM experimented with the stacking of another layer of 100 memory units. The hyperparameters on which the stacked LSTM experimented are given below:

- **Memory units -** 100 + 100(stacking an additional layer).
- **Steps (How many)** - Experimented with - 5- 30 window size.
- **Activation function -** ReLu (Rectified Linear Unit).
- **Optimizers** - Adam.
- **Loss** - MSE (Mean Squared Error) .
- **Epochs** - Experimented with 500 - 1000 epochs.

- **Result after Experimentation:**

Stacking of layers added the depth in the model but the number of parameters increases as compared to the LSTM. After experimentation, the hyperparameters for which the stacked LSTM converges and gives good results are epoch as 600 and step-size at 5.

4.4.3 Bi-Directional LSTM

BiLSTM can process the data from past to future and future to past so as to capture the dependencies in both directions.The hyperparameters on which the Bi-directional LSTM experimented are given below:

- **Memory units** - 50.
- **Steps -(How many)** - Experimented with : 5- 30 window size.
- **Activation function** - ReLu (Rectified Linear Unit).
- **Optimizers** - Adam.
- **Loss** - MSE (Mean Squared Error) .
- **Epochs** - Experimented with 500 - 1000.
- **Result after Experimentations:**

After experimentation, the hyperparameters for which the BiLSTM started converging are epochs at 550 and window step size of 5. Here, memory units are 50 which is standard for a large dataset, as the 50 units are assigned to the forward layer and 50 units are assigned to the backward layer. Remaining hyperparameters are untouched as changing them will degrade the performance.

4.4.4 Temporal Fusion Transformer (TFT)

Temporal fusion transformer is a multi-step time-series forecasting algorithm but in this study, single step forecasting is considered for comparative analysis. The hyperparameters for the TFT model for time-series forecasting of single step are given below:

- **Prediction length** - 1 (TFT is multi-horizon, we can predict next n points).
- **batch_size -** 128.
- **Loss function -** MSE.
- **Learning rate -** 0.03.
- **Attention heads -** (for large dataset 4 heads).
- **Dropout -** 0.1.
- **Epochs -** 100 -500 epochs.
- **Result after Experimentations:**

After experimentation with different hyperparameters, the model converges at epochs 150, attention heads 4 and dropout rate 0.1. The attention heads can be increased further, but it results in an increased number of parameters and the model becomes bulkier.

4.5 Evaluation Matrices

In this study there are three different evaluation matrices used for the analysis of a time-series forecasting.

4.5.1 MSE (Mean Squared Error)

Mean squared error is the standard model evaluation metric which is mainly used in the continuous value prediction task such as regression, time series forecasting etc. It measures the average error between the actual and predicted error. It penalizes the large errors with the squaring heavily so that it is helpful while dealing with large errors.

$$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n}\left(y_t - \hat{y}_t\right)^2$$

Where,

$\hat{y}_t -$ *represents Predicted value*

$y_t -$ *represents Actual value*

$n -$ *represents total data points*

4.5.2 SMAPE (Systematic Mean Absolute Percentage Error)

SMAPE [18] is mainly used to calculate the percentage error, as it was observed that during calculation of percentage error using the MAPE (Mean Absolute Percentage Error) formula, the denominator value may become zero, then the percentage error output is Infinite. The MAPE(Mean Absolute Percentage Error) is formulated as follows,

$$\text{MAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{y_t - \hat{y}_t}{y_t}$$

The term y_t represents the actual value at time t. If y_t=0 for any time point, the corresponding term in the MAPE calculation becomes undefined or tends toward infinity, potentially distorting the overall error metric. To solve this issue, SMAPE is the best choice as it handles the denominator becoming closer to zero.

$$\text{SMAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|y_t - \hat{y}_t|}{\left(|y_t| + |\hat{y}_t|\right)/2}$$

Where,

$y_t -$ *represents actual value*

$\hat{y}_t -$ represents predicted value at time t ,

$n -$ represents total number of observations

$|y_t - \hat{y}_t| -$ represents absolute error between actual and predicted values

$\frac{|y_t| + |\hat{y}_t|}{2} -$ represents average of the absolute actual and predicted values

4.5.3 RMSE (Root Mean Squared Error)

RMSE is the square root of the MSE. RMSE introduced so as to make the error scale compatible with the original data. It is mainly used when the units of the dataset matters. It is more sensitive to the outliers. In this study, the dataset consists of sales in terms of units so that RMSE is used along with MSE to analyze the error of models which is compatible with sales units.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(y_t - \hat{y}_t\right)^2}$$

Where,

$\hat{y}_t - represents\ Predicted\ value$

$y_t - represents\ Actual\ value$

$n - represents\ total\ data\ points$

This chapter has described different time-series data analysis methods, trend, seasonality analysis, and experimentation with LSTM and its variants along with a transformer based model. The next chapter aims to define inventory optimization problems and perform demand data generation to support the development and evaluation of the optimization model.

Chapter 5

INVENTORY OPTIMIZATION WITH DEMAND FORECASTED DATA

This chapter presents the optimization problem definition, description of objective function which represent the mathematical formulation for the optimization problem and demand data generation algorithms for realistic demand data.

5.1 Optimization Problem Definition

5.1.1 Problem Definition

The objective is to develop a robust inventory optimization model that leverages demand forecasting to maintain optimal stock levels in a supply chain system. The model aims to ensure product availability to meet customer demand while minimizing overall costs and inventory related risks such as stockouts, overstocking, and product wastage.

5.1.2 Objectives

- Optimize the ordering of medicine/surgical equipment by considering the constraints reordering point, current demand, safety stock, current stock, and the next day demand.
- Improve space utilization by factoring in lead time, holding costs, and storage capacity.
- Reduce inventory holding costs while ensuring medicine availability.
- Enhance supply chain efficiency by integrating supplier lead times into the inventory model.

5.2 Objective Function

The objective function which is mathematical representation of the optimization problem is given below,

$$Z = \min \sum_{i=1}^{n} (h_i \cdot I_i + c_i \cdot O_i + (E_d \cdot P_f))$$

Where,

Z = Total cost of inventory for i type of medicines/surgical equipment
h_i = Holding cost of medicine i (constant)
I_i = Average inventory level of medicine i
c_i = Ordering cost of medicine i (constant)
O_i = Quantity of medicine i ordered
P_f = Penalising Factor

Here, Z - total cost of inventory should be minimized by considering the constraints like space, daily budget and reducing the stock-out condition.

Objective function contains some important formula that needs to calculated on the fly during the calculation along with the constraints for optimization function are described below,

5.2.1 Inventory Stocks Considering Demand And Pending Orders

Inventory updates every day based on the received the pending orders, deduction of the demand for that medicine.

$$I_i = I_i^{\text{Current}} + Q_i^{\text{Pending}} - D_i$$

Where,

I_i^{Current} = The current stock of medicine i
Q_i^{Pending} = Pending orders of medicine i (ordered but not yet received)
D_i = Demand for medicine i
I_i = Average inventory level of medicine i

- **Reordering Point (ROP):**

The reordering point is the threshold point at which the order should be placed for a particular product/medicine. The reordering point formula used in this study is modified

by replacing the average demand by the average of forecasted demand for n days. This modification makes the reordering point as a dynamic reordering point for the given medicine/product.

$$ROP_i = D_i \cdot L_i + SS_i ROP_i = D_i \cdot L_i + SS_i$$

Where,

$ROP_i =$ Reorder point for medicine i
$D_i =$ Average demand for medicine i over n days
$L_i =$ Lead time for medicine i (in days)
$SS_i =$ Safety stock for medicine i

- **Safety Stock Formulation:**

Safety stock[19] formula used in this optimization function is the standard formula, which is mainly used in industry inventory management.

$$SS = k \cdot \sqrt{(L \cdot \sigma_D^2) + (D^2 \cdot \sigma_L^2)}$$

Where,

$SS =$ Safety stock
$k =$ Service level factor (e.g., 1.65 for 95% service level)
$\sigma_D =$ Standard deviation of daily demand (demand variability)
$L =$ Average lead time (in days)
$\sigma_L =$ Standard deviation of lead time (lead time variability)
$D =$ Average daily demand

5.2.2 Constraints

- **Ordering Quantity Constraint:**

The ordering quantity which is the decision variable should be greater than the deduction of current inventory stock from reordering point.

$$O_i >= \max\left(0,\ ROP_i - \left(I_i^{\text{Current}} + Q_i^{\text{Pending}}\right)\right)$$

- **Demand Constraint:**

The inventory should have enough stock to fulfil the demand, such that the current stock of inventory should be greater than the demand after deduction of safety stock.

$$I_i \geq D_i - SS_i$$

- **Ordering Constraint:**

Pending orders and inventory should be positive integers.

$$Q_i \geq 0 \quad \text{and} \quad I_i \geq 0$$

- **Stockout Definition:**

It is the undesirable condition where the current inventory runs out of stock. The E_d is Exceeding demand at the time of stock-out state.

$$E_d = \max(0,\ D_i - I_i)$$

5.2.3 Decision Variable

O_i is the order quantity that will be placed based on the requirement and considering the forecasted data. The order quantity is a decision variable, it means that the optimization function solves the problem and gives the optimized order quantity based on the forecasted data and constraints.

$$O_i = \text{Quantity of medicine } i \text{ ordered}$$

5.3 Realistic Demand Data Generation

The optimization problem introduced in the 5.2 section requires the realistic demand data for simulation of the results. The realistic data should have some criteria like it follows the seasonality, stationarity, trend etc. Specific criteria for realistic demand generation is discussed below.

5.3.1 Criteria For Realistic Demand Data Generation:

- Weekly sales of a medicine should have some pattern or sales should be high on some days, like weekends or working days.
- Monthly sales of a medicine should have some seasonality like, a medicine related to cold, infection or cough etc will have higher sales in the winter season, similarly for summer season the medicine related to dehydration, heat related etc will have higher demand.
- Medicines in off season should have low sales and the data should also have some noise in days between.

5.3.2 Customized Algorithm for Generating Medicine Demand

Algorithm 1 Generate Seasonal Medicine Demand Data

Initialize DataFrame with dates from `start_date` to `end_date`
for each date d **do**
    Extract `Month`, `DayOfWeek` (0 = Monday), and `Week` number
    if Month $\in \{12, 1, 2\}$ **then**
        Season $\leftarrow$ `winter`
    else if Month $\in \{3, 4, 5, 6\}$ **then**
        Season $\leftarrow$ `summer`
    else
        Season $\leftarrow$ `Monsoon`
    end if
end for
for each date d **do**
    Generate base demand:
      Medicine_A $\sim \mathcal{N}(80, 10)$
      Medicine_B $\sim \mathcal{N}(60, 8)$
      Medicine_C $\sim \mathcal{N}(70, 5)$
      Medicine_D $\sim \mathcal{N}(50, 10)$
end for
Define seasonal multipliers:
  `winter`: {A: 1.5, B: 0.6, C: 1.0, D: 0.8}
  `summer`: {A: 0.6, B: 1.4, C: 1.0, D: 1.3}
  `Monsoon`: {A: 0.9, B: 1.0, C: 1.0, D: 1.0}
for each date d **do**
    Multiply base demands by seasonal multiplier
end for
for each date d **do**
    if DayOfWeek $\in \{0, 5, 6\}$ **then**
        Increase all medicine demands by 20%
    end if
end for
for each date d **do**
    for each medicine with seasonal preference **do**
        if Season $\neq$ peak season **and** random chance $< 10\%$ **then**
            Add random noise $\sim$ Uniform(10, 30)
        end if
    end for
end for
for each medicine on each date **do**
    Add noise $\sim \mathcal{N}(0, 0.05 \times \text{demand})$
    Ensure demand ≥ 0
end for
return DataFrame with: `Date, Month, Week, Medicine_A, Medicine_B, Medicine_C, Medicine_D`

- **Algorithm : Generate Medicine Demand Data**

This algorithm simulates realistic, seasonal demand data for different types of medicines over a specified time period. It incorporates seasonal variations, weekly demand spikes, randomness, and domain-specific logic to closely mimic real-world demand fluctuations.

1. Objective:

The algorithm is designed to generate realistic and dynamic seasonal demand data for medicines over a specific time period.

2. Date Initialization:

A DataFrame is initialized with a sequence of dates ranging from the start date to the end date.

3. Feature Extraction:

For each date, the algorithm extracts:

- Month
- Day of the Week
- Week Number

4. Season Classification:

Based on the month:

- Winter: December, January, February
- Summer: March, April, May
- Monsoon: All other months

5. Base Demand Generation:

For each date, the base demand for the four medicines is generated using normal distributions, where each demand value is a normally distributed random variable denoted by X~ N(μ,σ2) X, with μ representing the mean demand and σ^2 the variance, capturing expected demand levels and their variability, respectively.

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

- Medicine A ~ N(80, 10)
- Medicine B ~ N(60, 15)
- Medicine C ~ N(70, 5)
- Medicine D ~ N(50, 10)

6. Apply Seasonal Multipliers:

Each medicine's demand is scaled by a season-specific multiplier, reflecting how seasonal factors influence consumption.

7. Weekend Demand Spike:

If the day is Friday, Saturday, or Sunday, all medicine demands are increased by 20% to simulate weekend sales spikes.

8. Random Disruptions for Off-Season Demand:

If a medicine is not in its preferred season, there is a 10% random chance that additional demand noise (Uniform(10, 30)) is added.

9. Add Small Gaussian Noise:

To further mimic real-world variability, Gaussian noise with mean 0 and standard deviation proportional to the demand is added.

10. Ensure Non-Negative Demand:

Any negative demand values are clipped to zero to maintain data validity.

11. Final Output:

The DataFrame presents the output generated by the customized demand-generation algorithm:

	Date	Month	Day	Medicine_A	Medicine_B	Medicine_C	Medicine_D
0	2025-01-01	1	1	137	36	67	34
1	2025-01-02	1	2	137	27	72	67
2	2025-01-03	1	3	138	70	78	43
3	2025-01-04	1	4	140	27	114	31
4	2025-01-05	1	5	122	28	85	50

Figure 5.1: Result of customized demand generation algorithm.

5.3.3 Observation of Customized Algorithm

The realistic data generation with the above mentioned algorithm which is a customized algorithm for realistic data generates the following results for 365 days. The graph represent the overall data points in week-wise and month-wise manner for analyzing that given algorithm able to generate the data that follows the criteria for realistic demand data generation,

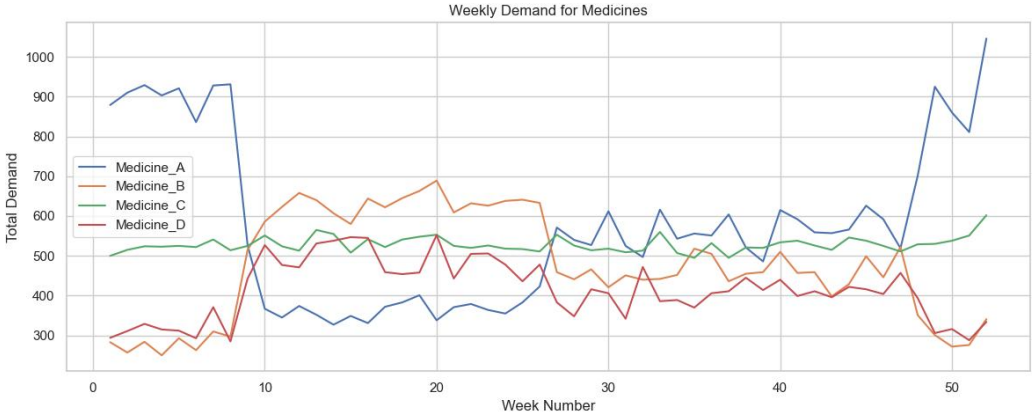

Figure 5.2: Generated data with customized algorithm 1

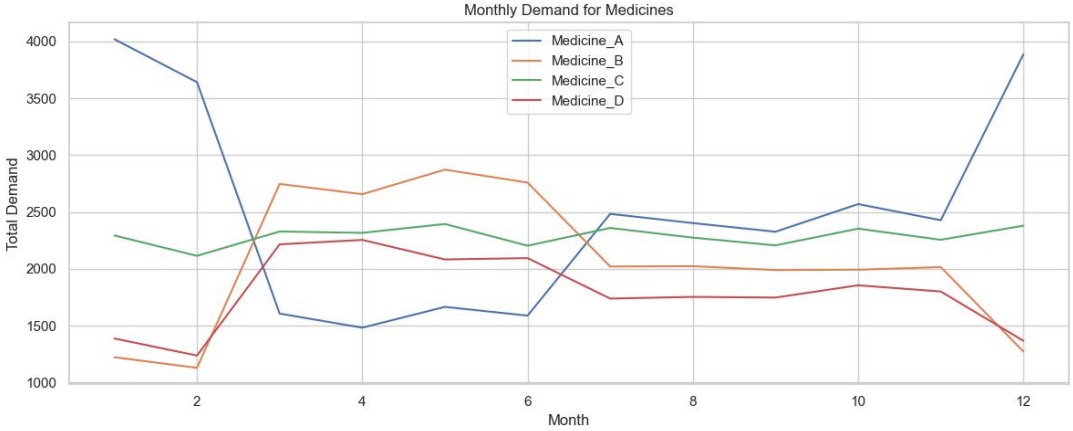

Figure 5.3: Generated data with customized algorithm 2

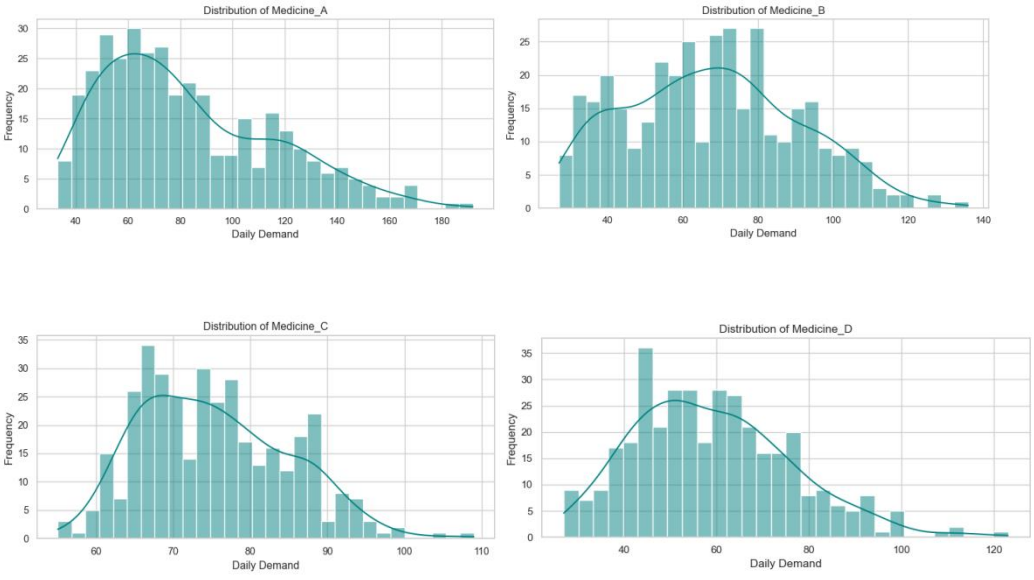

Figure 5.4: Data distribution of generated data

- **Observations**

1. In fig.5.1 the data generated by the customized algorithm have the seasonality, for e.g. medicine A and B are showing the seasonal peak of demand and other medicines are stationary in nature.
2. The data generated for medication C and D are stationary, it means there is no seasonal peak, no increasing or decreasing trend.
3. In fig.5.3 the distribution of data is analyzed, medicines B and D follow the normal distribution whereas the medicines A and C are slightly skewed towards the right.
4. The data generated by the customized realistic demand data generation algorithm follows most of the criteria for good realistic data generation.

5.4 Time GAN For Data Generation

Time GAN is used for time series data generation. The working of the time GAN is discussed in chapter 3. The hyperparameters used in Time GAN for demand data generation are given below:

- **Module** - GRU(Gated Recurrent Unit).
- **Hidden_dim** - 24.
- **Z_dim** - 32.
- **Gamma** - 1.0.
- **Batch size** - 128.
- **Learning Rate** - 0.001.
- **Iteration** - 10000.
- **Sequence length** - 300.

5.4.1 Model Training And Results

Time GAN is trained over the customized algorithm that is introduced in this study. Graphs against the Real vs Synthetic data represents the overall result of the GAN prediction over the original data. The generated results of this algorithm are given below,

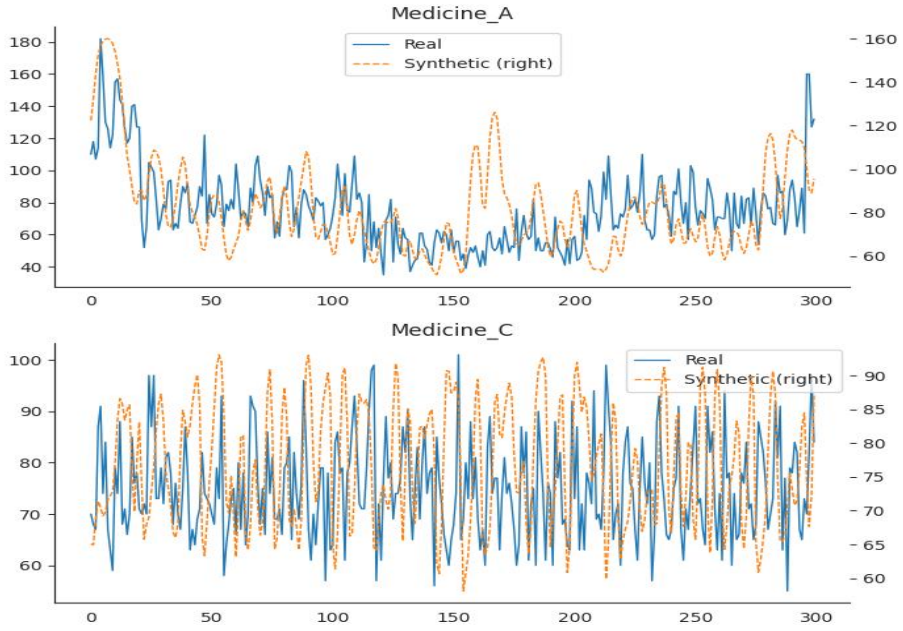

Figure 5.5: Generated data with Time GAN 1

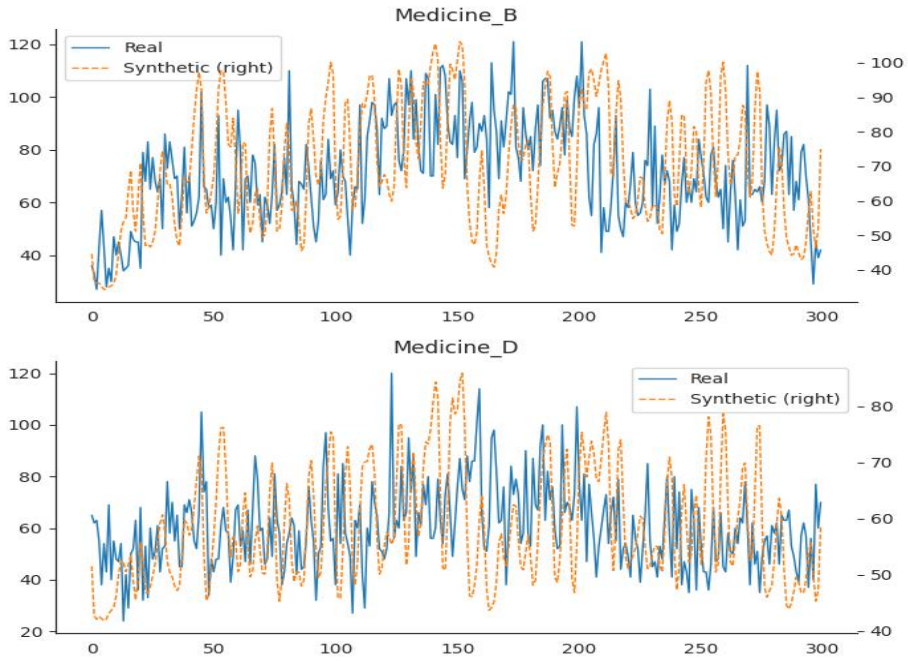

Figure 5.6: Generated data with Time GAN 2

- **Observations:**

1. Medicine_A shows a decreasing trend followed by a slight increase in demand, which is strongly learned by the GAN.
2. For Medicine_B and Medicine_D, the GAN reflected the general level of fluctuation observed in the real data.

3. Time GAN struggles with seasonality of the short term events and also the demand volatility and sudden spikes are not able to be captured properly.
4. The overall results of the GAN based demand generation are as per expectations. The generated data follows the most of the underlying criteria for realistic demand data generation.

5.5 Optimization Problem Solving With Pulp

Pulp is a Python [20] library which is mainly used for solving the optimization problems specifically, Linear, Mixed integral programming etc.

The following are some key characteristics of the PuLP library:

- High-Level LP Modeling Tool.
- PuLP allows you to define decision variables, objective functions, and constraints in Python using a readable, mathematics-like syntax.
- You can define both: Continuous variables (for LP) and Integer/binary variables (for MIP/ILP)
- Solver Interface (Not a Solver Itself). PuLP is just a modeling interface. It sends the model to a solver like CBC (default and included), GLPK, CPLEX (commercial), Gurobi (commercial).

5.5.1 Optimization On Sample Data

- The lead time is constant for any medication so that the standard deviation of lead time is 0.
- Current stock is the current level of inventory.
- Pending orders are the orders that have been placed earlier but not received because the orders are in transit.
- Predicted demand is the forecasted demand for the given medicine.
- Holding cost is based on the per-unit basis, the unit cost of Medicine A is 0.3, it means it is 2.5 % of the unit cost of a medicine.
- Unit cost represents the cost of a medicine.

Sr.no	Name	Current Stock (units)	Pending Orders (units)	Predicted Demand (units)	Holding Cost (Rs.)	Unit Cost (Rs.)	Lead time (Days)
1	Medicine A	40	10	30	0.3	12	2
2	Medicine B	60	15	50	0.2	10	2

Table 5.1: Input sample data for optimization problem

In this section, the input sample data consists of both constants and variables. The constants include holding cost, unit cost, lead time, and budget. The specific constant values for Medicine A and Medicine B are provided in Table 5.1. The variables comprise current stock, pending orders, predicted demand, average demand, safety stock, and the reorder point. The generated demand data for Medicine A and Medicine B is used for the calculations, with predicted demand obtained using the Stacked LSTM model. The values for current stock and pending orders are dynamically computed to ensure that all imposed constraints are satisfied.

5.5.2 Optimized Result

Optimized order quantity is calculated by solving the optimization function given below,

$$Z = \min \sum_{i=1}^{n} (h_i \cdot I_i + c_i \cdot O_i + (E_d \cdot P_f))$$

Where,

Z = Total cost of inventory for i type of medicines/surgical equipment
h_i = Holding cost of medicine i (constant)
I_i = Average inventory level of medicine i
c_i = Ordering cost of medicine i (constant)
O_i = Quantity of medicine i ordered
P_f = Penalising Factor

The components required to solve the above optimization function are calculated as outlined below:

1. Inventory Level (Ii)

$$I_i = I_i^{\text{Current}} + Q_i^{\text{Pending}} - D_i$$

I_i^{Current} = The current stock of medicine i

Q_i^{Pending} = Pending orders of medicine i (ordered but not yet received)

D_i = Demand for medicine i

I_i = Average inventory level of medicine i

Table 5.1 provides all the necessary values current stock, pending orders, and predicted demand required to compute the inventory levels for Medicine A and Medicine B.

Medicine A : $I_i = 40 + 10 - 20 = 30$ units.

Medicine B : $I_i = 60 + 15 - 50 = 25$ units.

2. Holding Cost ($h_i \times I_i$)

Given Holding Cost per Unit:

Medicine A: Rs. 0.3

Medicine B: Rs. 0.2

Total Holding Cost:

Medicine A: .3×30=Rs. 9.0

Medicine B: 0.2×25=Rs. 5.0

3. Ordering Cost (c_i x O_i)

Given Unit Price (used as Ordering Cost):

Medicine A: Rs. 12.

Medicine B: Rs. 10.

Ordering cost is calculated based on the optimized order quantity O_i value will vary based on the LPP solution.

4. Exceeding Demand (Ed)

For both Medicine A and Medicine B, the value of E_d (exceeding demand) is zero, indicating that there is sufficient stock available to meet the forecasted demand. The penalizing factor P_f is defined as the unit price of the respective medicines.

Since E_d=0 units. for both medicines, the penalty cost term: $E_d * p_f = 0$. This implies that no additional penalty cost is incurred due to stockouts in the current scenario for either medicine.

5. Safety Stock

$$SS = k \cdot \sqrt{(L \cdot \sigma_D^2) + (D^2 \cdot \sigma_L^2)}$$

$SS =$ Safety stock
$k =$ Service level factor (e.g., 1.65 for 95% service level)
$\sigma_D =$ Standard deviation of daily demand (demand variability)
$L =$ Average lead time (in days)
$\sigma_L =$ Standard deviation of lead time (lead time variability)
$D =$ Average daily demand

Here, the service level factor k is set to 1.65, corresponding to a 95% service level. Since the lead time is constant, the standard deviation of lead time is zero. The standard deviation of daily demand is calculated using the forecasted demand and average demand data provided in Table 5.1, which simplifies the overall safety stock calculation.

k = 1.65

σL = 0.

L = 2 days.

Medicine A: Safety Stock = 11 units..

Medicine B: Safety Stock = 14 units.

6. Dynamic Reordering Point

$$ROP_i = D_i \cdot L_i + SS_i$$

$ROP_i =$ Reorder point for medicine i
$D_i =$ Average demand for medicine i over n days
$L_i =$ Lead time for medicine i (in days)
$SS_i =$ Safety stock for medicine i

Medicine A: Reordering Point = 71 units.

Medicine B: Reordering Point = 114 units.

Constraints

Daily budget <= Rs. 1000.

$$O_i >= \max\left(0,\ ROP_i - \left(I_i^{\text{Current}} + Q_i^{\text{Pending}}\right)\right)$$

Medicine A: O_i >= max(0, 21) i.e O_i >= 21 units.

Medicine B: O_i >=max(0,39) i.e O_i >= 39 units.

The optimization function integrates all essential components—including holding cost, ordering cost, inventory level, exceeding demand, penalizing factor, reorder point, and safety stock (calculated dynamically) to determine the optimal order quantity for each medicine. This objective function is formulated as a *Linear Programming Problem (LPP)* and is solved using the *PuLP* library in Python, which offers robust tools for modeling and efficiently solving such optimization problems.

The optimized order for the given medication ,

Optimized Order	**Medicine A**	**Medicine B**
	22 units	60 units

Table 5.2: Optimized order quantity

The optimized results indicate that an order quantity of **22 units** for **Medicine A** and **60 units** for **Medicine B** should be placed to achieve optimal inventory management.

This chapter describes the optimization problem definition, objective function representation, demand data generation with customized and GAN algorithms. The next chapter focuses on the results and analysis of the deep learning algorithms used for demand forecasting.

Chapter 6

RESULT AND ANALYSIS

6.1 Comparative Analysis of Algorithms

This chapter discusses the results of the deep learning and Transformer models and analysis based on the evaluation matrices MSE, SMAPE and RMSE etc.

The MSE values of each algorithm used for demand forecasting is given below,

ATC Class	LSTM	Stacked LSTM	Bidirectional LSTM	TFT
M01AB	8.14	8.06	8.13	22.33
M01AE	4.64	4.70	4.62	4.63
N02BA	4.50	4.46	4.74	11.32
N02BE	152.30	154.35	155.15	205.95
N05B	18.21	18.42	18.27	125.37
N05C	1.23	1.23	1.22	2.01
R03	62.06	62.46	64.16	12.60
R06	4.95	5.06	5.08	6.09

Table 6.1: MSE values of all algorithms.

The SMAPE values of each algorithm demand forecasting task of 8 ATC classes given below,

ATC Class	LSTM	Stacked LSTM	Bidirectional LSTM	TFT
M01AB	0.47	0.47	0.47	0.50
M01AE	0.47	0.45	0.47	0.47
N02BA	0.60	0.55	0.61	0.65
N02BE	0.34	0.31	0.34	0.34
N05B	0.43	0.40	0.43	0.50
N05C	1.60	1.59	1.58	1.58
R03	0.92	0.93	0.93	0.80
R06	0.59	0.59	0.59	0.59

Table 6.2: SMAPE values of all algorithms.

Analysis based on table 6.1 and 6.2 are given below

6.1.1 LSTM

- Generally performs well across most categories, achieving the best MSE in simpler tasks such as M01AB, M01AE, and N05C .
- Struggles with more complex categories like N02BE(152.30) and R03(62.06) compared to TFT.

6.1.2 Stacked LSTM

- Marginally improves over basic LSTM in some cases (M01AE, N05C) but performs similarly or slightly worse overall.
- Similar issues as LSTM for complex categories.

6.1.3 Bidirectional LSTM

- Shows slight improvements over the basic LSTM in categories like N02BA and R03.
- Its bidirectional nature helps capture dependencies better than standard LSTM.

6.1.4 Temporal Fusion Transformer (TFT)

- The Temporal Fusion Transformer outperforms the other models significantly in categories like R03 (MSE = 12.60 compared to 79.14 for LSTM) and N02BA (MSE = 4.3 vs. 4.6 for LSTM).
- Performs worse in cases like M01AB (22.33 vs. 8.14 for LSTM) and N02BE (205.95 vs. 152.30 for LSTM).
- Mixed performance, excelling in complex patterns but struggling in some cases.

6.2 Poor Performance Of TFT

- TFT requires substantial data to train effectively.
- Univariate data lacks these additional inputs, reducing TFT's ability to leverage its architecture for feature interactions.
- TFT is designed to incorporate known future variables (e.g. holidays, promotions, weather).
- TFT performs better with multivariate time series data as compared to Univariate data.

6.3 Comparative Analysis

- LSTM, Stacked LSTM, and Bidirectional LSTM show relatively stable RMSE values across most ATC (Anatomical Therapeutic Chemical) classes.
- N02BE and R03 medication classes contain extreme values and their distribution of data is right skewed.
- TFT struggles with complex patterns (e.g., N02BE, N05B) but performs well for some cases (e.g., R03).
- Stacked LSTM often achieves the lowest RMSE compared to other LSTM based models.
- For consistent performance across most classes, Bidirectional LSTM is the best choice.
- For predicting the next 'n' sales, TFT is a better option over the other LSTM derivatives.

Here, this chapter describes the comparative analysis of each algorithm based on the evaluation metrics values. The next chapter highlights the conclusion of this study, limitations of current work, and future work to enhance the project.

Chapter 7

FUTURE WORK AND CONCLUSION

7.1 Conclusion

- The study of Deep Learning algorithms for demand forecasting suggests that the LSTM based algorithms perform better than the transformer based models.
- LSTM and its variants perform better overall on all classes but no single algorithm is best for all the classes. Stacked LSTM achieved a better SMAPE score than other variants.
- The TFT algorithm is the state-of-the-art model for multi-step time series forecasting, but it requires a sufficient amount of data to perform effectively. Since the size of our dataset is small and the study focuses on univariate single-step time series forecasting, the TFT model struggles to perform well.
- Demand data generation using customized algorithms and Time GAN gives good results as they follow all criteria for effective demand data generation.
- The optimization function solved using the LPP with the help of the PULP library gives logical results for the generated data of a medication.

7.2 Limitations

The following are the limitations of the current work:

- The size of the dataset is small hence many algorithms didn't perform up to the mark.
- This study focuses on next-day demand forecasting, as multi-step forecasting using LSTM and its variants is generally less effective. These models are primarily suited for one-step forecasts and tend to struggle with multi-step predictions.
- Solving the optimization problem with generated data does not validate the performance of the optimization function with original data.
- The dataset is univariate but in the real world there are many factors that are responsible for demand forecasting.

- Optimization function should be customizable as there are many different factors in inventory management which influence overall cost and storage requirements.

7.3 Future Works

There are many areas where the project could be enhanced and upgraded. Some scenarios and suggestions are provided below,

- Multivariate time series data can be explored for demand forecasting as it will have some extra features which will be helpful in better prediction.
- Multi-step forecasting with the help of some state of the art models like Informer, Temporal Fusion transformer can be explored for better results.
- This project mainly uses the daily sales for demand forecasting but hourly, weekly, monthly huge time series data can be experimented for further patterns and important features extraction.
- Generalized optimization function which will solve the inventory optimization for major industries can be explored with the help of industry experts.
- Integration of demand forecasting and inventory optimization can be done with the web application and ML based service which mainly solves the inventory optimization problems.

BIBLIOGRAPHY

[1] R. Gangwar, “Crafting a Future-Ready Supply Chain with Advanced Inventory Optimization Strategies,” International Journal of Research and Analytical Reviews, vol. 10, pp. 185–194, Dec. 2023.

[2] “Research on Optimization of Inventory Management Based on Demand Forecasting,” Applied Mechanics and Materials, vols. 687–691, pp. 4828–4831, Nov. 2014.

[3] S. İmece and Ö. F. Beyca, “Demand Forecasting with Integration of Time Series and Regression Models in Pharmaceutical Industry”, JEPS, vol. 34, no. 3, pp. 415–425, 2022, doi: 10.7240/jeps.1127844. https://doi.org/10.7240/jeps.1127844

[4] R. Siddiqui, M. Azmat, S. Ahmed, and S. Kummer, "A hybrid demand forecasting model for greater forecasting accuracy: the case of the pharmaceutical industry," Supply Chain Forum: An International Journal, vol. 22, no. 2, pp. 124-134,Sep.2021.doi:10.1080/16258312.2021.1967081.https://doi.org/10.1080/16258312.2021.1967081

[5] G. Emmanuel, Y. Arifin, I. Sonata and M. Zarlis, "Drug Demand Prediction Based on Epidemiology Factors Using Random Forest," 2023 15th International Congress on Advanced Applied Informatics Winter (IIAI-AAI-Winter), Bali, Indonesia, 2023, pp. 1-6, doi:10.1109/IIAI-AAI-Winter61682.2023.00010.10.1109/IIAI-AAIWinter61682.2023.00010

[6] M. S. Amalnick, N. Habibifar, and M. Hamid, "An intelligent algorithm for final product demand forecasting in pharmaceutical units," International Journal of System Assurance Engineering and Management, vol. 11, pp. 481–493, Apr. 2020. doi: 10.1007/s13198-019-00879-6.https://doi.org/10.1007/s13198-019-00879-6

[7] R. Rathipriya, A. A. Abdul Rahman, S. Dhamodharavadhani, et al., "Demand forecasting model for time-series pharmaceutical data using shallow and deep neural network model," Neural Computing and Applications, vol. 35, pp. 1945–

1957, Jan. 2023. doi: 10.1007/s00521-022-07889-9 https://doi.org/10.1007/s00521-022-07889-9

[8] A. Ouaddi and J. Benhra, "Transformer-Based Model Applied to Customer Demand Forecasting: A Comparative Analysis," 2024 Sixth International Conference on Intelligent Computing in Data Sciences (ICDS), Marrakech, Morocco, 2024, pp. 1–8, doi: 10.1109/ICDS62089.2024.10756471.

[9] E. Brophy, Z. Wang, Q. She, and T. Ward, "Generative Adversarial Networks in Time Series: A Systematic Literature Review," ACM Computing Surveys, vol. 55, no. 10, pp. 1–31, Oct. 2023, Art. no. 199, doi: 10.1145/3559540.

[10] Na. Li,Donald M. Arnold et al. "From demand forecasting to inventory ordering decisions for red blood cells through integrating machine learning, statistical modeling, and inventory optimization," Transfusion, first published Nov. 16, 2021. [Online]. Available: https://doi.org/10.1111/trf.16739

[11] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, “Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting,” arXiv preprint arXiv:1912.09363, 2020. [Online]. Available: https://arxiv.org/abs/1912.09363

[12] J. Yoon, D. Jarrett, and M. van der Schaar, “Time-series Generative Adversarial Networks,” in Advances in Neural Information Processing Systems, vol. 32, 2019. [Online].Available:https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf

[13] M. Zdravkovic, “Pharma sales data,” Kaggle, 2022. [Online]. Available: https://www.kaggle.com/datasets/milanzdravkovic/pharma-sales-data

[14] S. Pandian, “Time Series Analysis: Definition, Components, Methods, and Applications,” Analytics Vidhya, Oct. 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-to-time-series-analysis/

[15] S. Lee, “Lag Operator Simplified: 5 Proven Methods for Time-Series Mastery,” NumberAnalytics,Mar.18,2025.[Online].Available:https://www.numberanalytics.com/blog/lag-operator-simplified-5-methods-timeseries-mastery

[16] J. Brownlee, “How to Check if Time Series Data is Stationary with Python,” Machine Learning Mastery, Aug. 15, 2020. [Online]. Available: https://machinelearningmastery.com/time-series-data-stationary-python

[17] D. Botache, K. Dingel, R. Huhnstock, A. Ehresmann, and B. Sick, “Unraveling the Complexity of Splitting Sequential Data: Tackling Challenges in Video and Time Series Analysis,” arXiv preprint arXiv:2307.14294, Jul. 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2307.14294

[18] Permetrics Documentation, “Symmetric Mean Absolute Percentage Error (SMAPE),”[Online].Available:https://permetrics.readthedocs.io/en/latest/pages/regression/SMAPE.html.

[19] E. Thieuleux, “Safety Stock Formula & Calculation: The 6 Best Methods,” AbcSupplyChain,Apr.7,2025.[Online].Available:https://abcsupplychain.com/safety-stock-formula-calculation/

[20] K. Keyush, “Linear Programming & Discrete Optimization with PuLP,” AnalyticsVidhya,Apr.7,2022.[Online].Available:https://www.analyticsvidhya.com/blog/2022/03/linear-programming-discrete-optimization-with-pulp/